How to Convert Between ALEPH Standard Formats of Bibliographic Records

Document Version: 1.0
Last Update: January 5, 2009

Table of Contents

1 OVERVIEW 5

2 EXAMPLES FOR THE INTEGRATION OF THE FIX_DOC_CONVIT-ROUTINE: CONVERTING MARC 21 TO MAB2 6

3 CALL AND FUNCTION OF THE CONTROL PROGRAM 7

4 USING THE CONVIT FUNCTIONS 8

4.1 Entries in the convit Table 8

4.2 Selecting Field Text 10

4.3 Generally Usable convit Functions 11

4.3.1 General field editing –'edit_field' 11
4.3.2 Conditional Processing of edit_field – 'checkit_const' and 'checkit_contains' 13
4.3.3 Translating Text First – 'cut_transl' 14
4.3.4 Generating Target Field Codes – 'cyclic_fieldno' 15
4.3.5 Defining Text for the Target Field – 'const_field' 16
4.3.6 Appending Text to a Target Field – 'str_cat' 16
4.3.7 Joining subfields - 'get_sub_all' 17

5 STRUCTURE OF THE CONVIT-TABLE 18

5.1 Standard Processing of a Table Row 20

5.2 Parameter Value for Selecting a Field Text 21

6 DESCRIPTION OF GENERALLY USABLE CONVIT FUNCTIONS 27

6.1 chkit_const–Checks by Comparing a Selected Text with Constant Text 28

6.2 chkit_contains–Checking the Appearance of Constant Text in Selected Text 31

6.3 const_field - Creating a Field with Constant Field Text 35

6.4 cut_transl–Cutting and Translating of Selected Text 36

6.5 cyclic_fieldno–Generation of Cyclic Field Numbers 38

6.6 edit_field–General Field Editing 40

6.7 str_cat–Appending Text to Existing Field 46

6.8 get_sub_all–Joining of All/Selected Subfields to a Single Subfield 48

7 DESCRIPTION OF MARC 21 TO MAB CONVIT-PROGRAMS 49

7.1 usm2mab_ldr–Processing MARC field LDR (Leader) 50

7.2 usm2mab_006–Processing MARC field 006 51

7.3 usm2mab_007–Processing MARC Field 007 52

7.4 usm2mab_008–Processing MARC field 008 53

7.5 usm2mab_440–Processing MARC field 440 54

7.6 usm2mab_700–Processing MARC field 700 (Authors) 55

7.7 usm2mab_710–Processing MARC field 710 (Corporate body) 57

8 DESCRIPTION OF THE UNIMARC-TO-MAB CONVIT-PROGRAMS 59

8.1 uni2mab_100–Processing UNIMARC fields 100, ... 60

8.2 uni2mab_210–Processing UNIMARC field 210 (Publishers) 61

8.3 uni2mab_225–Processing UNIMARC field 225 (Common Title) 63

8.4 uni2mab_500–Processing UNIMARC field 500 64

8.5 uni2mab_501–Processing UNIMARC field 501 65

8.6 uni2mab_503–Processing UNIMARC field 503 66

8.7 uni2mab_700–Processing UNIMARC field 700 (Authors) 67

8.8 uni2mab_710 – Processing UNIMARC field 710 (Corporate body) 68

9 DESCRIPTION MAB-TO-MARC CONVIT-PROGRAMS 69

9.1 mab2usm_002–Processing mab-field 002 (Date) 70

9.2 mab2usm_037–Processing mab-field 037 (Language Codes) 71

9.3 mab2usm_050–Processing mab-fields 050, 051, ... 72

9.4 mab2usm_100–Processing mab-fields 100, 104, ... (Authors) 73

9.5 mab2usm_200–Processing mab-fields 200, 204, ... (Corporate body) 74

9.6 mab2usm_425–Processing mab-field 425 (Year of Publication) 75

9.7 mab2usm_451–Processing mab-fields 451, 461, ... (Common Title in Original Form) 76

9.8 mab2usm_540–Processing mab-fields 540, 541, ... (Standard Book Numbers) 77

9.9 mab2usm_902–Processing mab-fields 902, 907, ... (Subjects).....78

9.10 mab2usm_700–Processing mab-field 700 (Notation).....80

9.11 mab2usm_800–Processing mab-fields 800, 802, ... (Secondary Entries)....81

1 Overview

This document describes the fix_doc_convit program. You can think of fix_doc_convit as a control program that offers a bundle of commands, functions, or programs for editing and converting the bibliographic records. It is written in a way that fits into the concept of Aleph's fix routines for editing bibliographic records. These routines are defined in the table tab_fix in the tab directory of the appropriate library. Fix_doc_convit is just another program you can assign to some fix routines. For an example of the integration of fix_doc_convit, see sections 2 Examples for the Integration of the fix_doc_convit-routine: Converting MARC 21 to MAB2 and 3 Call and Function of the Control Program.

Which function of fix_doc_convit are applied to which field of the bibliographic record is defined in a specific table, the 'convit-table'. The structure of the 'convit-table' is explained in section 4.1 Entries in the convit Table.

> **Note**: An important concept of the fix_doc_convit function, called text selection, is explained in section 4.2 Selecting Field Text.

The functions of fix_doc_convit can be categorized as follows:

- Functions for general use
- Special functions that serve some dedicated tasks for the conversion between the commonly used formats in librarianship (e.g. for the conversion from MARC 21 to MAB2).

See section 4.3 Generally Usable convit Functions to get an overview of the functions for general use and some examples on how to use them.

The rest of this document, called the Reference Section describes each function and concept in detail.

2 Examples for the Integration of the fix_doc_convit-routine: Converting MARC 21 to MAB2

In this example we use fix_doc_convit to convert records in MARC 21 format via Z39.50 into MAB format “on the fly”.

$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            USMAB
...
```

$data_tab/tab_fix (ext04):

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

So, for the purpose of converting to MAB format you define a fix routine “USMAB” which consists of running the fix_doc_convit control program. What actions are actually performed during the conversion of a record is given in the convit table that is specified in the ‘FILE=’-parameter, tab_fix_usm2mab in this case. The fix_doc_convit control program reads the convit-table and controls the conversion by calling the appropriate conversion functions listed in the convit table.

3 Call and Function of the Control Program

As shown in the section above, you have to set an entry in the table ‘tab_fix’ of the corresponding library in order to use the "fix_doc_convit" control program. The exact structure of this entry is as follows:

$data_tab/tab_fix:

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

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

Fffff Identifier of the fix-routine

- Common identifiers are:
  `USMAB – MARC 21 to MAB`
  `UNMAB - UNIMARC to MAB`
  `MABUS - MAB to MARC 21`
  `MABDC – MAB to Dublin Core`
  `DCMAB – Dublin Core to MAB`
  `USMDC – MARC21 to DC`
  etc.

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

- Structure – see section 5 Structure of the convit-table
- Common names for the corresponding format conversions are:
  `tab_fix_usm2mab   MARC21 to MAB`
  `tab_fix_uni2mab   UNIMARC to MAB`
  `tab_fix_mab2usm   MAB to MARC21`
  `tab_fix_mab2dc    MAB to Dublin Core`
  `tab_fix_usm2dc    MARC21 to Dublin Core`
  `tab_fix_usm2qdc   MARC21 to qualified Dublin Core`
  etc.

-- mandatory parameter--

DEFAULT= Indication of the standard convit-function for processing a field when the column program/parameter in the convit-table is empty.
If no DEFAULT= is given here, the standard convit function is automatically set to the convit function ‘edit_field’ (see section 6.6 edit_field–General Field Editing)

4 Using the convit Functions

4.1 Entries in the convit Table

The table consists of five 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
```

Typically, in Col.1 and Col.2 you specify the field from the source record whose content has to be edited and put into a field of the target record with the field code given in Col.3 and Col.4. The way the content is edited is given with the convit function specified in Col.5.

Common masking characters ("#" and "!") may be used for the specification of the field codes.

Examples

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

- Transfer of the LDR-field of the source record to the target record (field code) remains the same.

- Transfer of the text of field 051 as subfield a of field 085
- Transfer of the field 100 with any indicator
- Transfer of the subfield b from field 200a to subfield c of field 910
- Transfer of the subfield a from field 331 with any indicator, field code remains the same
- Transfer of the subfield a from field 200a to subfield a of field 910
- Transfer of all 441-fields with any indicator as field 910 with indicator identical to the first position of indicator of the source field, i.e. field 441ab becomes 910a, as well as field 441ac.

4.2 Selecting Field Text

Some convit functions often require the selection of a certain section of the field text as a parameter, for example, to test it against some value or to copy it (and not the rest of the field text) into a new target field.

Several aspects have to be considered for the field text selection:

- Code of field/subfield
- Specification of a certain occurrence of a field (for example, in case of a repeatable field)
- The field used for selection is coming from source or target record
- Position and length of a text section in the field text

To formulate these aspects of field text selection, a certain syntax has been developed to be used with the convit functions.

Field text selection generally consists of two parts, first

- Selection of a field

then

- Selection of text within this field

fieldtext-selection **Is** ***text-selection***
or ***text-selection/field-selection***
or ***/field-selection***

This notation means that the specifications following the / character define the field selection part of a field text selection; the specifications in front of the / define the text selection part.

If no field selection is specified, the text selection takes place on the field text of the actual field by default.

Here are some examples of possible field text selections. Refer to the detailed description of field text selection in the reference section of this document.

The first example returns just one character

```
[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

The following examples show some advanced options of field text selection

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

- Field selection: the last but one occurrence of field 328 (because of {L2}) irrespective of indicator (##) from the so far produced fields of the target record (:OUTP)
- Text selection: 1 character starting as position 6 form the last ({L}) subfield a.

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

- Field selection: the 2nd ({2}) occurrence of field 528 irrespective of indicator (##) from source record (:I).
- Text selection: Subfield text of last subfield ({L}).

4.3 Generally Usable convit Functions

This section explains the functionality of some generally usable functions that can be used in field conversion tasks. It gives an overview of the characteristics of the function and gives you examples on how to use them. For an exact description of each function, refer to the Reference Section of this document.

4.3.1 General field editing –'edit_field'

This function realises the basic functionality for the adoption of field and subfield contents from the source record to the target record. Therefore, this program is used as the default function (unless you specified another). For example, instead of writing the following:

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

You can write:

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

or even shorter:

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

because ADDNEW as the specified action is the default parameter of the edit_field function. The effect is that you just transfer the text from a field in the source record to a field in the target record.

Example:
If the convit table states

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
001   a 001
```

you get from field
001 $$aM295012897X
of the source record the following field in the target record:
001 M295012897X

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
089   a 24500 n edit_field             MERGE-I
…
331#  a 24500 a edit_field             MERGE-I
333   a 24500 c edit_field             MERGE-I
335#  a 24500 b edit_field             MERGE-I
…
359   a 24500 c edit_field             MERGE-I
```

…

This example with “action” indicator MERGE-I describes how the target field 24500 and its subfields are composed from several fields of the source record. Therefore, if the source records contain the fields:

```
331  $$aHow the university works
335  $$ahigher education and the low-wage nation
359  $$aMarc Bousquet
```

you get:

```
24500 $$aHow the university works$$bhigher education and the low-wage
nation$$cMarc Bousquet
```

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
418#  a 260     edit_field ADDNEW,SEL=ag,RENAME=aagb
```

This example transfers subfields a and g (SEL=ag) of 418 to subfields a and b (RENAME=aagb) of 260. (If you use SEL= and RENAME= an indication of subfields in column 2 or 4 has no meaning.)

Example:

```
418   $$aLondon$$gHarrison
```

You get

```
260  $$aLondon$$bHarrison
```

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
425#  a 260   c edit_field MERGE-I,FIRST=Y,CAT=";cc, ;"
```

This example adds a subfield c to 260. FIRST=Y indicates that, if there are already some target fields existing in the target record, the first one found is taken for processing. If there are several 425 fields, their contents are merged in subfield c 260 using a comma (,) as a delimiter.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
130## a 304   a
130## l 304   a edit_field MERGE-I,PREFIX="<",SUFFIX=">",CAT=";aa ;"
```

Build subfield a of 304 from all subfields l to 130 and put “<…>” around it.

Example:

```
130 0$$aThree little pigs$$lEnglish
```

You get:

```
304  $$aThree little pigs <English>
```

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
028## a 425   a edit_field          ADDFIRST
030## a 425   a edit_field          ADDFIRST
031## a 425   a edit_field          ADDFIRST
```

The action-indicator ADDFIRST causes only one the source fields 028, 030, or 031 to be written to the target field 425. If field 028 is not empty, it is transferred to the target field. If field 030 is not empty, it is transferred to the target field only if the target field is still empty. Otherwise field 030 is discarded.

Example:

```
30  $$a1965
31  $$a1978
```

You get:

```
425  $$a1965
```

4.3.2 Conditional Processing of edit_field – 'checkit_const' and 'checkit_contains'

These functions implement a conditional processing of the edit_field function based on the comparison of a field text selection to a certain text string referred to as constant text. If the comparison results in a true value, the edit_field function is performed as described above. Therefore, all the parameters of the edit_field function can be used, plus some extra parameters that define the text comparison. The most important of these extra parameters are the following:

CHK-SRC= Specification of source text to be compared using fieldtext selection according to section 5.2 Parameter Value for Selecting a Field Text. Using the mechanism of fieldtext selection, the source of the text is totally independent form the source or target fields specified in cols. 1-4 of the convit table. It may be derived from any other field of the input record or from the fields generated so far.

CHK-OP= Relational operator for text comparison, such as:

- `EQ` – equal
- `NE` – not equal
- …

(Only applicable in case of chkit_const)

CHK-TXT= Constant text as 2.operand for comparison
In case of chkit_contains, the constant text can contain wildcards.

Example:

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
003## a 700a  a chkit_contains       CHK-SRC=b,CHK-TXT=UDC
003## a 700b  a chkit_contains       CHK-SRC=b,CHK-TXT=DDC
```

For this sequence, perform the following:
If subfield b of field 003 contains the text string UDC, transfer the contents of field 003, subfield a to subfield a of field 700, indicator a; otherwise, do nothing with field 003. Then, check if it contains the text string 'DDC'. In this case, transfer the contents of 003, subfield a to 700, indicator b. (If subfield b of 003 contains UDC as well as DDC, the first transfer is lost.)

Example:

003 $$a823.912$$bDDC

You get:

700b $a823.912

4.3.3 Translating Text First – 'cut_transl'

This function generates the target field text by first selecting text with field text selection from the source field text, translating it, and then performing any processing the edit_field function offers.

The extra parameters for the cut_transl function are:

FROM= Specification of the source text to be selected and translated with field text selection according to section 5.2 Parameter Value for Selecting a Field Text
-- mandatory--

TO= Specification of target for new text with field text selection according to section 5.2 Parameter Value for Selecting a Field Text

- Only specification of ***text-selection*** in the form of ***sf***, [***pos***:***len***] and their combinations are taken into account.
- This means the new text is stored as a subfield or a fixed field text.
- If necessary, starting at position ***pos*** in full length order up to maximum length ***len.***

Default= fixed field text from position 0

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

Processing of the TRANSL parameter

The TRANSL parameter consists of several elements, the translation definitions, which are separated by a certain delimiter character, the translation definition delimiter. In the example below, it is the semicolon (;). But if the semicolon itself is part of a translation definition, you can choose any other suitable character as a delimiter.

A second form of delimiter, the string definition delimiter, separates the source and target string definition of a translation definition. Therefore, a translation definition has the following form:

Source string definition – Delimiter – Target string definition

This means that if the selected source field text matches the source string definition, it is replaced by the target string definition.

This delimiter is defined in the second position of the TRANSL parameter; in the example below it is the "/"-character, but you can use any other suitable character.

Because the string definition delimiter is defined in the second position of the TRANSL parameter and is also already interpreted as part of the first translation

definition, the source string definition is always empty. Therefore, this translation definition applies if source field text selection results in an empty string.

Example:

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
LDR##   014     cut_transl FROM=[6:1],TO=a,TRANSL=";/;m/computer file;/;"
008##   010     cut_transl FROM=[35:3],TO=a
```

The first example selects the character at position 6 of LDR. If it is an “m”, the target field 014 is generated with the text string “computer file”. The TRANSL parameter consists of three translation definitions: The first just consists of a backslash (/) and applies if source field text selection delivers an empty string. Because the target string definition is also empty, no text for the target field is generated in this case.

The second translation definition is a “regular” translation definition that writes the string “computer file” to the target field if the source field text selection equals “m”.

The third translation definition, also just consisting of a backslash “/”, is a special form of translation definition. It applies if source field text selection is not empty and does not equal “m”. Because the target string definition is empty, no text for the target field is generated. If this last translation definition is missing, the unaltered source field text selection, i.e. the character at position 6 from the source field, is written to the target field.

The second example just shows some “substring” functions.

Example:

```
LDR## 00000nms^^22001697^^4500

831117|1978^^^^is^||||^||||||^|||^^heb|
```

You get:

```
014  $$aComputer file
010  $$aheb
```

4.3.4 Generating Target Field Codes – ‘cyclic_fieldno’

This function combines the functionality of edit_field with the automatic generation of target field codes.

The most important extra parameters concerning the target field code generation are:

CYC-ADD= Numerical value for generating the new target field code. The new target field code is obtained by adding the value of CYC-ADD to the existing target field code
- 1 <= ***nnnn*** <= 500

-- mandatory --

CYC-CNT= Number of iterations of addition.
- 1 <= ***nnnn*** <= 500

-- mandatory --

Example:

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
002## a 100   a cyclic_fieldno       CYC-ADD=004,CYC-CNT=025
```

In this example, the first occurrence of source field 002 goes to target field 100, the second to 104, the third to 108 and so on, 25 times at the most.

Example:

```
002  $$aWilson, Roman
002  $$aMorgan, Rita
```

You get:

```
100 $$aWilson, Roman
104 $$aMorgan, Rita
```

4.3.5 Defining Text for the Target Field – 'const_field'

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

The problem is that this new target field is usually created only once per record, independent of 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 than zero. If a target field for this ONCE-IDN-Value of the current record has already been created, the function is terminated without effect.

Example:

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
#####   LDR     const_field   "00000nM2.01200024||||||h",ONCE-IDN=01
#####   030     const_field   "zz|||||||||17",ONCE-IDN=02
#####   050     const_field   "||||||||a|||||",ONCE-IDN=03
#####   FMT     const_field   "MH",ONCE-IDN=04
```

This creates the fields LDR, 030, 050, and FMT once in every target record, independent of any source field.

4.3.6 Appending Text to a Target Field – 'str_cat'

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

Example:

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
245## a 331   a
245## n 331   a str_cat ", "
245## p 331   a str_cat ": "
```

This section works as follows:

First transfer text from source field 245 subfield a to target field 331 subfield a, if possible. Then add text from 245 subfield n to 331 subfield a again, if possible. If text has already been generated for 331 from the previous step, use a comma (,) as a separator text between the existing text and the text coming form 245 subfield n. Accordingly, the text from 245 subfield p is processed.

Example:

245 $$aDissertation abstracts.$$nA$$pThe humanities and social sciences

You get:
331 $$aDissertation abstracts, A: The humanities and social sciences

4.3.7 Joining subfields - 'get_sub_all'

This function combines the texts of all or selected subfields of the source field into one single text of the target field.

Example:

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
600##   711c  a get_sub_all "/",-67
```

This function call generates a string for Subfield a of target field 711, indicator c, from the subfields of source field 600. The first parameter "/" indicates a delimiter string. This means that by assembling the target text the individual parts coming from the appropriate subfields are separated by the string "/".

The second parameter describes the subfield selection. By default, all subfields are incorporated, but by specifying a subfield selection you can restrict the function to certain subfields. For example, if you write "acd", only subfields a, c, and d are incorporated. If you precede the subfield selection with a "-"sign, all subfields are incorporated besides those listed. Therefore "-67" as in the example above, means take all subfields. All subfields are taken except subfield 6 and subfield 7.

Example:

60010$$a Pushkin, Aleksandr Sergeevich,**$$d**1799-1837**$$x**Museums**$$z**Russia (Federation)**$$z**Moscow**$$v**Maps$$61234

You get:

700c $$a Pushkin, Aleksandr Sergeevich/1799-1837/Museums/Russia (Federation)/Moscow/Maps

5 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

- Specification of a subfield-code of the source field
- The precise meaning of this information depends on the convit-function in column 5.
- Usually, this means that this subfield is chosen (the first one if there is a multiple appearance). The remaining subfields are not needed and are deleted.

Col-3 - Target-Field-Code

- Specification of a field code with indicator for a new target field.
- The precise meaning of this information depends on the convit-function in column 5.
- Normally, this means the field name of the new field. If the Target-Field-Code contains the "#"-character. These positions are usually be replaced with the corresponding characters from the source field.

Col-4 - Target-Subfield-Code

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

Col-5 - Function and Parameter

- Specification of a convit-function and possibly related parameters.
- There has to be at least a blank between function name and parameter.
- Parameters without a function name are not allowed. The string from the first character that is unequal to a blank up to the next blank or the end of row is interpreted as a function name.
- If this column is empty, the convit-function that is defined under DEFAULT= is processed (by default: "edit_field").
- The text following the function name is interpreted as a list of parameters.
- The parameters of the list are separated by commas.
- A parameter given by position, i.e a parameter without an introductory key word with an equals sign, must always be placed at a given position in the list.
- A parameter with an introductory key word and an equals sign may be put at any position in the list.
- The value of a parameter may be put between quotation marks (Cobol-QUOTE-sign). It **must** be put between quotation marks, if:
  - Blanks at the beginning and/or at the end are part of the value
  - Commas or equals signs are part of the value; otherwise, those characters are interpreted as classifying elements of the list.
  - If the quotation marks are part of the value, a prefixed backslash is necessary. The backslash itself has to be coded with double backslashes.
- By a hexadecimal indication of a character in form of **\x*XX*** (whereas ***XX*** are two hexadecimal characters 0-9,A-F). In the value, any character can be entered.
- By a hexadecimal indication of a Unicode-value in form of **\u*XXXX*** (whereas ***XXXX*** is four hexadecimal characters 0-9,A-F). For the value, any UTF8-Code of a Unicode-value can be entered.
  You cannot perform a global translate to UTF8 with the created record, because the record contains these signs; the current character-set of the current record must be UTF8 when working from fix_doc_convit.
- A special case of a parameter value is the selection of field-text. For a more comprehensive description, see section 5.2 Parameter Value for Selecting a Field Text.

5.1 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=.

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
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*** is generated. Positions with the masking character "#" are 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 is transferred.

Structure of the new field text

- Source-Subfield-Code ***x*** = empty, Target-Subfield-Code ***y*** = empty
  - The field text is transferred 1:1.
- Source-Subfield-Code ***x*** = not empty, Target-Subfield-Code ***y*** = empty
  - The subfield text of the first matching subfield is the new field text.
  - If no Target-Subfield-Code is specified, the subfield text is transferred **without** the subfield label and therefore, the new field is a plain field without a subfield structure.
  - If no subfield ***x*** can be found or the subfield text is empty, no new field is generated.
- Source-Subfield-Code ***x*** = not empty, Target-Subfield-Code ***y*** = not empty
  - The subfield text of the first matching subfield ***x*** is transferred 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 are generated.
- Source-Subfield-Code ***x*** = empty, Target-Subfield-Code ***y*** = not empty
  - The whole field text is transferred 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 plain field without subfield structures.
  - If the old field text is empty or has a subfield structure, no new subfield and consequently no new field are 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 an indicator identical to the first position of the indicator of the source field; the field text is adopted 1:1.

5.2 Parameter Value for Selecting a Field Text

The parameterization of a convit function often requires the specification of a certain text area of a doc-sequential record, for example, to copy it or to check it.

Usually, one or more of the following specifications may be necessary:

- Subfield code
- OCCUR-designation for the subfield
- Relative position within a subfield text or fixed field text
- Maximum length as from the relative position
- Field code
- OCCUR- designation 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
- 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

We have created a syntax which can include all specifications mentioned above. Whether the function in question actually uses all of them depends on the function itself. The function also ignores redundant indicators or writes an error to the logfile, if necessary, as well as selects default values if specifications are missing.

If a parameter allows the selection of field text in this way, it is mentioned in the parameter description of the function, for example, "Selection of field text according to ...", "***fieldtext-selection***", "***fieldtext-sel***".

In the following section is a description of the general syntax, including some examples.

General Rules:

- The individual elements can be separated with one or more blanks to improve readability.
- "Default" means the way a function 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 specification of a subfield code.

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

field-selection Selection of a field from a doc-record

- This happens before the selection of text.
- Because a text selection is typically always specified and field selection is the exception, this combination has been chosen.

Default= The data of the "actual field" is used for text selection.

text-selection **::=** ***Sf***
or **[** ***pos*** **:** ***len*** **]**
or **[** ***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 is 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 is 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* the occurrence of the subfield in the complete selected field text is used.
Default= The first field found with *fieldcode* is used.

ALL
FIELDTEXT

The complete field text is selected, regardless of whether the field has a subfield structure or is a plain 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* is used.

field-record Specification of record for field selection

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

`O` | `OUT` | `OUTPUT` - Field is 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 is 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

Some possible values for ***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 the end.
`LNF` `LAST-NOT-FIRST`	-	The last but not the first object
`LNF`*pppp* `LAST-NOT-FIRST`*pppp*	-	The pppp'th object from the end but not the first object.

Examples:

```
[6:1]/008
```

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

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

- Field selection: The second to last occurrence of field 328 irrespective of the indicator from the fields produced so far of the new output record
- Text selection: 1 character starting at position 5 from the last subfield a.

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

- Field selection: The second occurrence of field 528 irrespective of the indicator from the input record.
- Text selection: Subfield text of last subfield.

6 Description of Generally Usable convit Functions

Overview

6.1	chkit_const	-	Checks by comparing a selected text with constant text
6.2	chkit_contains	-	Checks the appearance of constant text in selected text
6.3	const_field	-	Creates a field with constant field text
6.4	cut_transl	-	Cuts and translating of selected text
6.5	cyclic_fieldno	-	Generates cyclic field numbers
6.6	edit_field	-	General field editing Standard program if no convit-program specified (if no other program is defined as standard by DEFAULT=)
6.7	Str_cat	-	Appends text to an existing field
6.8	get_sub_all	-	Merges all/selected subfields into one subfield

6.1 chkit_const–Checks by Comparing a Selected Text with Constant Text

This function compares selected field text with constant text. If the comparison is successful/true, the processing continues with the standard program edit_field. If the comparison is unsuccessful/negative/false, the processing of the table row is cancelled 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 come 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 section 6.6 edit_field–General Field Editing)
, 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 section 5.1.Standard Processing of a Table Row.
- If the comparison is successful, the processing continues with program edit_field.

CHK-SRC= Specification of source text to be compared uses field text selection according to section 5.2 Parameter Value for Selecting a Field Text.
If the parameter WHICH= is used, the 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.operand) and constant text (2.operand)

- `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.operand for comparison
-- mandatory --

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

Default= N

WHICH= Specification of keyword for determination of fields given in CHK-SRC= that is tested 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 also the last applicable field (this means: there are at least two applicable fields)
`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 at least two applicable fields)

Default= utilization of ***field-occur***-specification from field text selection; if o value is there, the first applicable field is used.

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

NOSF-OK= **WHICH=ALL**:
If there is no subfield to be found according to ***text selection***, the testing of this field is false by default and therefore the complete result is also false. With NOSF-OK=Y, the testing of this field is 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 is cancelled. With NOSF-OK=Y, the result is assessed as true and processing continues.
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 therefore the complete result is also false. 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 is 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 considered, the result of the testing is false and processing of the field is cancelled. 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 is assessed as true and processing continues.

Default= N

6.2 chkit_contains–Checking the Appearance of Constant Text in Selected Text

This function 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 is continued with edit_field, otherwise the processing of the table row is cancelled without effect.

- The field text selection is determined and tested by **one** field if:
  - No field selection with **CHK-SRC=**, the field text from the actual buffer is used.
  - Field selection with **CHK-SRC=**, but without **WHICH=**ALL or **WHICH=**ONE-OF-ALL
- The field text selection is determined and tested 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 non-existence 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= exists and the search text is contained in the selected text
  - The complete result can be modified 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= exists in **at least one** field and search text is contained in the 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 is rue:
    - Selected text according to *text-selection* in CHK-SRC= exists and the search text is contained in the selected text
    - Does a subfield according to *text-selection* in CHK-SRC= does not exist, NOSF-OK=Y has to be specified.
    - Does a substring according to *text-selection* in CHK-SRC= does 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 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 come from the input record or from the fields already generated. 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 section (6.6 edit_field–General Field Editing)
, 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 columns 1 – 4 is equal to the standard described in section 5.1 Standard Processing of a Table Row.
- If the comparison is successful the processing continues with program edit_field .

CHK-SRC= Specification of source text to be compared using field text selection according to section 5.2 Parameter Value for Selecting a Field Text
If the parameter WHICH= is used, the OCCUR specification has no meaning for the field selection and is therefore not allowed in this context.
-- mandatory --

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

WHICH= Specification of keyword for determination of fields given in CHK-SRC= that is tested 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 also the last applicable field (this means: there are at least two applicable fields)
`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 at least two applicable fields)

Default= utilization of ***field-occur***-specification from field text selection; if that field has no value, the first applicable field is used

WILD= Y/N-Switch

The characters star "*****" and question mark "**?**" in the search string is 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 to 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 is cancelled. With NOFLD-OK=Y, set this yields to a true result and processing continues.

Default= N

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

If there is no subfield to be found according to ***text selection***, the testing of a certain field is false by default and therefore the complete result is also false. With NOSF-OK=Y, the testing of this field is 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 is cancelled. With NOSF-OK=Y, the result is assessed as true and processing continues.

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 therefore the complete result is also false. 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 is 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 considered, the result of the testing is false and processing of the field is cancelled. 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 is assessed as true and processing continues.

Default= N

6.3 const_field - Creating a Field with Constant Field Text

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

There is a problem that this new target field is normally created only once per record, independent of the Source-Field-Code *CCCc*. 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 than zero. If a target field for the ONCE-IDN-Value of the current record has already been created, the program is terminated. By specifying the Source-Field-Code as ##### the creation of the target field is

- Independent of a source field and is always processed
- Created right after the first cycle through the table
- Created first, if this is the first entry in the table

If you specify the same ONCE-IDN-Value during several program calls, you can make the creation of the target field with the constant text dependent on 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 section 5.1 Standard Processing of a Table Row.
- The indication of the subfield in column 2 is irrelevant for the text selection; the specified field-text is taken as “chosen” text

field-text Specification of the constant target field text
- If column 4 is used by target subfield-code, this text may not contain any subfield label.
- Otherwise, the text may contain one or several subfield labels, but the proper formal structure is not 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“

6.4 cut_transl–Cutting and Translating of Selected Text

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

The new field text, if any is present, is further processed with edit_field. Therefore, all parameters for edit_field can be used in addition to the parameters described here.

The effect is that the field text from the field given in column 1 is replaced in the working area and further 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 **edit_field** (see section 6.6 edit_field–General Field Editing)
, FROM=*fieldtext-selection*
, TO=*fieldtext-selection*
, CASEIGN=*sw*
, TRANSL=*transl-str*

- The meaning of column 1 – 4 is equal to the standard described in section 5.1 Standard Processing of a Table Row.
- Is the new field text not empty the processing continues with edit_field.

FROM= Specification of the source text to be cut out and translated with field text selection according to section 5.2 Parameter Value for Selecting a Field Text.
-- Mandatory--

TO= Specification of target for new text with field text selection according to section 5.2 Parameter Value for Selecting a Field Text.
- Only specification of ***text-selection*** in the form of ***sf***, [***pos***:***len***] and their combinations are taken into account.
- This means the new text is stored as a subfield or a plain field text, if necessary starting at position ***pos*** in full length or 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 description see section 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 second 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 is 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 is 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 the default for “other” values.
  - If the target string is empty too, no field text is built.
- If the target string definition in the last but not first command is not empty and no appropriate source string definition could be found, the text selected with FROM= remains unchanged.

6.5 cyclic_fieldno–Generation of Cyclic Field Numbers

This function 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 continues with edit_field. Therefore, all parameters for edit_field can be used in addition to the parameters 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 is generated by repeated ("cyclic") additions of a certain amount to the source field code. This cyclic addition 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 field, the indicator can be incorporated.

This cyclic addition normally works on the three digits of the numeric source field code but can be restricted to just a part of this field code. The remaining part does not necessarily have to be 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 stops, even if the maximum number of possible repetitions is not used yet.

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

***parameter* :** same as for **edit_field** (see 6.6 edit_field–General Field Editing)
, 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 section 5.1 Standard Processing of a Table Row. Deviating from this column 2 is replaced with the generated target field code internally.
- If a new target field code can be generated, the processing continues with edit_field.

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 on 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**

6.6 edit_field–General Field Editing

This function transfers field- and subfield contents from the source document to the target document. This function is used as the default value for the parameter DEFAULT= of the control program, i.e. for the standard function with empty "program" column (col. 5) of the convit table.

This means that this function does not need to be entered into 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 in section 5.1 Standard Processing of a Table Row.
- The specification of the subfield in column 2 and 4 is irrelevant if the parameter SEL= or RENAME= is not empty.

action Type of transfer of the new field to the target-record:
The existence of a homonymous field in the new target record has to be considered. To determine this field, either the 3-digit field code of the new field without an indicator or the field code with an indicator is considered. The second option is used if the action ends with "**-I**".
By default, the last located field is used; if FIRST=Y it is 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 is discarded.

ADDNEXT, ADDNEXT-I
- Save new field at the end of the new record, if there are homonymous fields.
- Otherwise the field is discarded.

EXCHANGE, EXCHANGE-I

- Replace of the last (or first if FIRST=Y) homonymous field, if there is one.
- Otherwise the field is treated as in ADDNEW/ADDNEW-I

CHGNOADD, CHGNOADD-I

- Replace of the last (or first) homonymous field, if there is one.
- Otherwise the field is discarded.

MERGE, MERGE-I

- The field must contain subfields.
- Appends the subfields of the new field to the last (or first) homonymous field, if there is one.
- If the parameter CAT= is used, first the impacts of MERGE/MERGE-I takes effect before CAT=
- Otherwise the field is treated as in ADDNEW/ADDNEW-I.

MRGNOADD, MRGNOADD-I

- The field must have subfields.
- Append the subfields of the new field to the last (or first) homonymous field, if there is one.
- If the parameter CAT= is used, first the impact of MRGNOADD/MRGNOADD-I takes effect before the impact of CAT=.
- Otherwise the field is 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 checked here. If the current indicator does not match the indicators in SEL-I=, the processing of the field is stopped without effect.
You may create a “negative selection” by specifying a minus character at the first position. Assign *CCCcc* with *CCC##* then

- The first character after the minus character is used as separator for the indicator details.
- Single-character indicators 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:

Besides renaming indicators according to Source-/Target-Field-Code (column-1/column-3) another form of indicator renaming is possible by using this parameter. To make this indicator renaming effective, the source indicator should be preserved by assigning ***TTTtt*** with ***TTT*****##**
You can define several renamings with this parameter, each consisting of four characters. The first two characters are used for selection/verification of the indicator and the following two characters specify the new indicator.

- The first character of the complete definition string is used as separator for the different renaming definitions
- A renaming definition is always filled with blanks (to reach a length of four characters)
- 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 is 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:

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

- The field must have a subfield-structure.
- A Specification of source/target-subfield in column 2/4 is irrelevant if you use SEL= and is ignored.

Standard= No selection of subfield

RENAME= Renaming of subfields:

Specifications of pairs of subfields as a character string; the first character is the subfield-code of the source field; the second character is the subfield-code of the target field.

- The field must have a subfield-structure.
- The first character in the renaming pair can be a "**#**" character, i.e. all remaining subfields are then renamed; therefore, such a pair should always be at the end of the subfield specification.
- A Specification of source/target-subfield via column 2/4 is irrelevant if you use RENAME= and it is ignored.

Standard= No renaming

CAT-INP= Merging ("catenate") of subfields from the source field.
Multiple CAT-commands can be coded consecutively in this parameter, which are processed in the order of their specification. The structure of the CAT-INP= parameter is somewhat more complex and is described below.

- The field must have a subfield-structure.
- Unlike the CAT=-parameter, this merging 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 was build according to Source/Target-Subfield-Code (column-2/4) and the parameters SEL=, RENAME= and CAT-INP=, the Prefix and Suffix is inserted to all subfields or the plain field text.

- A Prefix/Suffix cannot be assigned specifically to a particular subfield. If you want to do so, then only this field may be generated.

CAT= Merging ("catenate") of subfields
Multiple CAT-commands can be coded consecutively in this parameter, which are processed in the order of their writing down. The structure of the CAT= parameter is somewhat more complex and is described below.

- 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 structure of the CAT-OUTP= parameter is somewhat more complex and is described below.

- The field must have a subfield structure
- Unlike CAT=-Parameter with *action* MERGE/MERGE-I all subfields is integrated equally in the merging-process
- This parameter only makes sense, if you work with *action* MERGE/MERGE-I

Standard= No merging of subfields

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

- The subfields is 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 are 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 are 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 already existent fields are part of the processing, depending on "action", the last matching field is used by default. Setting FIRST=Y causes the first located field to be chosen.

Standard= N

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

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

- The first character (semicolon, in the example above) defines the delimiter for the individual CAT-definitions and may be used for only this purpose in the CAT parameter string. If the semicolon is to be part of the separator string of a CAT-definition, another suitable character must be chosen as the delimiter.
- Each CAT-definition consists of the specification of a subfield-code-pair and a separator string. The separator string is built from the characters following the subfield code pair, up to the delimiter character (the semicolon). The separator string is placed between the two catenated subfields. The separator string can be an empty string.
- First CAT-definition: append all subfields-a to the **last** existing subfield-b. If there is no subfield-b, the first subfield-a is renamed to subfield-b and all remaining subfields-a and appended to it. The separator string is blank-slash-blank.
- Second CAT-definition: merge all subfields-c to only one subfield-c, i.e. all additional subfields-c is appended to the **first** subfield-c. The separator string is comma-blank.
- Third CAT-definition: analogue to the first CAT-command but without a separator string; the subfield texts are directly joined.
- Forth CAT-definition: append all remaining subfields to the last existing subfield-x. If there is no subfield-x, the first remaining subfield is renamed to subfield –x and all remaining subfields are appended to it. The separator is blank-colon-colon-blank. All subfields with a Subfield-Code that have not been used explicitly in earlier CAT-definitions are the remaining subfields.
- Fifth CAT-definition: subfields with the same Subfield-Code are merged to one subfield, analogous to the second CAT-command.

Attention

- At *action* MERGE/MERGE-I the CAT-parameter is 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 are always included in the target search but never used as source-subfield.
- Therefore, 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.

6.7 str_cat–Appending Text to Existing Field

This function appends 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 is generated in a way as if the program-/parameter-column would be empty and the standard processing takes place.

This program should be used for fields with subfield-structure only and in conjunction with a Specification of a source/target-subfield-code.

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

parameter* : *delimiter

- The meaning of columns 1-4 is equal to the standard described in section 5.1 Standard Processing of a Table Row, if no existing new field according to ***TTTtt*** can be located.
- Additionally, the Target-Field-Code can be used for target field search. Therefore, 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

Depending on Source- and Target-Subfield ***x***/***y*** and of the field text structure of the Source- and Target-Field, the following new field text in the last matching target field is built:

- The entire field text (in case of the source subfield code ***x*** is empty even if field has subfield structure) or the subfield text of the first subfield ***x*** is used as the text to append. If no text has been determined (field text empty, or no subfield ***x*** located), the processing is cancelled.
- The source text is appended to the entire field text of the target field (in case of target subfield code ***y*** is empty or target field has no subfield structure) or to the first located subfield ***y*** with ***delimiter***. If no subfield ***y*** can be located, a new subfield ***y*** is created at the end of the target field text.
- In fields with subfield structure Source- and Target-Subfield-Code ***x***/***y*** should always be coded; otherwise, it does not make sense and may lead to erroneous results.

6.8 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 is placed between the source texts. Otherwise the texts are written together directly.

The order of the source field persists and is not changed according to the selective indication ***subf-sel***.

With this text, a new field ***TTTtt*** is created, either as fixed field text or as subfield ***y*** (Target-Subfield-Code ***y*** = empty/not empty).

A source field without a subfield structure is not processed.

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

parameter **:** ***subf-sel***
, delimiter

- The meaning of columns 1and 3 is equal to the standard described in section 5.1 Standard Processing of a Table Row.
- The column 2 (Source-Subfield-Code) is irrelevant. A possible value is ignored.

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

Standard= all subfields

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

7 Description of MARC 21 to MAB convit-Programs

Overview

7.1 usm2mab_ldr - Processing MARC-field LDR (Leader)

7.2 usm2mab_006 - Processes MARC-field 006

7.3 usm2mab_007 - Processes MARC-field 007

7.4 usm2mab_008 - Processes MARC-field 008

7.5 usm2mab_440 - Processes MARC-field 440

7.6 usm2mab_700 - Processes MARC-field 700 (Authors)

7.7 usm2mab_710 - Processes MARC-field 710 (Corporate body)

7.1 usm2mab_ldr–Processing MARC field LDR (Leader)

This program converts the MARC-field LDR. From the information of this field the fixed MAB-fields "**050**", "**051**" and/or "**052**" are generated or corrected, if they already exists as new fields.

These fields are also treated by other usm2mab-programs.

A detailed description of the specific updates and dependencies, mostly relating to position, are not 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 is not analyzed in the program.
- Target-Field-Code, Source-Subfield-Code, and Target-Subfield-Code are not analyzed and are irrelevant.

7.2 usm2mab_006–Processing MARC field 006

This program converts the MARC-field 006. From the information of this field the fixed MAB-fields "**050**", "**051**" and/or "**052**" are generated or corrected, if they already exists as new fields.

These fields are treated by other usm2mab-programs, too.

A detailed description of the specific updates and dependencies, mostly relating to position, are not 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 is not analyzed in the program.

Target-Field-Code, Source-Subfield-Code, and Target-Subfield-Code are not analyzed and are irrelevant.

7.3 usm2mab_007–Processing MARC Field 007

This program converts the MARC-field 007. From the information of this field the fixed MAB-fields "**`050`**" and/or "**`057`**" are generated or corrected, if they already exists as new fields.

These fields are treated by other usm2mab-programs, too.

A detailed description of the specific updates and dependencies, mostly relating to position, are not 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 is not analyzed in the program.
- Target-Field-Code, Source-Subfield-Code, and Target-Subfield-Code are not analyzed and are irrelevant.

7.4 usm2mab_008–Processing MARC field 008

This program converts the MARC-field 008. From the information of this field the fixed MAB-fields "**050**" and/or "**051**" are generated or corrected, if they already exist as new fields.

These fields are also treated by other usm2mab-programs.

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

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

A detailed description of the specific updates and dependencies, mostly relating to position, are not 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 is not analyzed in the program.
- Target-Field-Code, Source-Subfield-Code, and Target-Subfield-Code are not analyzed and are irrelevant.

7.5 usm2mab_440–Processing MARC field 440

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

If a subfield $$v is included in the source field, the additional target field "**455**" is generated from 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 is not analyzed in the program.
- The Target-Field-Code is adopted 1:1 and must not include "#". If necessary, the indicator is modified by the program.
- Source-Subfield-Code and Target-Subfield-Code are not analyzed and are irrelevant.

7.6 usm2mab_700–Processing MARC field 700 (Authors)

This program processes the MARC-fields containing author's data, usually the fields "700" and "720".

The next free target field for MAB-author-fields "100"and "104" up to "196" is determined. The search starts at the Target-Field-Code of the table entry.

If all target fields already exists, the process is 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 is 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 is not analyzed in the program.
- The Target-Field-Code determines the search's starting section for the next, not yet existing, new field "100"and "104" to "196". The indicator also remains in the specific Target-Field-Codes.
- The Source-Subfield-Code is not analyzed and is 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 characters "`#`" and "`!`".
- Exsample: `LNG-TRG=A00b`

Default= A00

- The assignment of a source field and a field in original language is the result of subfield 6; for example, Sf-6 in the current author field 100##. Then the field with field code LANG-SRC= which has an identical subfield 6, is searched.
- The new text for the target field is built in the same manner as the source field.
- The target field codes are generated accordingly to "100", "104", ..., i.e. the second and third character (00, 04, ...) are taken over; the other characters come from LNG-TRG=.

7.7 usm2mab_710–Processing MARC field 710 (Corporate body)

This program processes the MARC-fields containing the corporation's data, usually the fields "710"/"711", but also "110"/"111".

The next free target field for MAB-corporate-fields "200" and "204" up to "296" is determined. The search starts at the Target-Field-Code of the table entry. If all target fields already exist, the process is 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 put in parenthesis 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 is not analyzed in the program.
- The Target-Field-Code determines the search's starting point for the next, not yet existing, new field "200" and "204" to "296". The indicator also remains in the specific Target-Field-Codes.
- The Source-Subfield-Code is not analyzed and is 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 a source field and a field in original language is the result of subfield 6; eg. Sf-6 in the current corporate body field 200##. Then the field with field code LANG-SRC= is searched, which has an identical subfield 6.
- The new text for the target field is built in the same manner as the source field.
- The target field codes are generated accordingly to “200” and “204”, ..., i.e. the second and third character (00, 04, ...) are taken over; the other characters come from LNG-TRG=.

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

Overview

8.1 uni2mab_100 - Processes UNIMARC fields 100, ...

8.2 uni2mab_210 - Processes UNIMARC field 210 (Publishers)

8.3 uni2mab_225 - Processes UNIMARC field 225 (Common title)

8.4 uni2mab_500 - Processes UNIMARC field 500

8.5 uni2mab_501 - Processes UNIMARC field 501

8.6 uni2mab_503 - Processes UNIMARC field 503

8.7 uni2mab_700 - Processes UNIMARC field 700 (Authors)

8.8 uni2mab_710 - Processes UNIMARC field 710 (Corporate body)

8.1 uni2mab_100–Processing UNIMARC fields 100, ...

The program processes the 100's UNIMARC fields. This information is 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 updates and dependencies, mostly relating to position, are not 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 according to the above description. Non-matching fields are not processed.
- "#####" could be entered to the first column, but "1####" narrows the program calls considerably. For optical/documentary reasons you can also create a specific line for each relevant field so that you do not forget a single relevant field.
- Target-Field-Code, Source-Subfield-Code, and Target-Subfield-Code are not analyzed and are irrelevant.

8.2 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 are created.

If there are new MAB-fields for the 1. Publisher but no fields for the 2. Publisher (neither 415 nor 417), these fields are created.

If there are fields for the 1. Publisher and for the 2. Publisher, a new repeatable field "418" is created based on the Publishers' data from the current UNIMARC field 210.

In addition, a new MAB-field "425" is created 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" is created once based on the first located data.
  - "**210##**" Subfield $$**d** ➔ "**425** " Subfield $$**a**
  - "**210##**" Subfield $$**h** ➔ "**425** " Subfield $$**a**

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
210##           usm2mab_210
```

- The Source-Field-Code has the standard meaning, but the current field code is not analyzed in the program.
- Target-Field-Code, Source-Subfield-Code, and Target-Subfield-Code are not analyzed and are irrelevant.

8.3 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"and "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 is cancelled. This also happens if the Source-Field does not contain a Subfield $$**a**.

From a standard number in Subfield $$x, the fields "452" and "462" to "492" are 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 $$**a**
- ***tt*** is taken from the Target-Field-Code of the table
- The Target-Subfield-Text is generated by merging the following Source-Subfield-Texts with delimiters
  - $$**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 is not analyzed in the program.
- The Source-Subfield-Code is not analyzed and is irrelevant.
- The Target-Field-Code determines the search's starting point for the next, not yet existing, new field "451" and "461" to "491". The indicator also remains in the specific Target-Field-Code. For the generated Target-Fields "452" and "462" to "492" always the indicator "**a**" is set.
- The Target-Subfield-Code determines the Subfield-Code for the new text of the fields "451" to "491". By default, $$a is used. Subfield $$a is always generated for the fields "452" to "492".

8.4 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 $$a has to be put in parenthesis even if there is no subfield $$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 is not analyzed in the program.
- The Target-Field-Code has the standard meaning, but it is not analyzed in the program.
- The Source-Subfield-Code is not analyzed and is irrelevant.
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. By default, $$a is used.

8.5 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 $$a has to be put in parenthesis even if there is no subfield $$a.

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

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

8.6 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 $$a has to be put in parenthesis even if there is no subfield $$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 is not analyzed in the program.
- The Target-Field-Code has the standard meaning, but it is not analyzed in the program.
- The Source-Subfield-Code is not analyzed and is irrelevant.
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. By default, $$a is used.

8.7 uni2mab_700–Processing UNIMARC field 700 (Authors)

The program processes the UNIMARC fields containing the authors data, usually the fields "700" and "720".

The next free Target-Field for MAB-author-fields "100" and "104" up to "196" is determined. The search starts at the Target-Field-Code of the table entry.

If all target fields already exist, the process is 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 determinants, the decoded text is 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 is not analyzed 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 analyzed and is irrelevant.
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. By default, $$a is used.

8.8 uni2mab_710 – Processing UNIMARC field 710 (Corporate body)

The program processes the UNIMARC fields containing the corporate bodies' data, usually the "710"field.

The next free Target-Field for MAB-corporate body-fields "200"and "204" to "296" is determined. The search starts at the Target-Field-Code of the table entry.

If all target fields already exist, the process is 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 put in parenthesis 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 is not analyzed in the program.
- The Target-Field-Code determines the search's starting point for the next, not yet existing, new field "200" and "204" to "296". The indicator also remains in the specific Target-Field-Codes.
- The Source-Subfield-Code is not analyzed and is irrelevant.
- The Target-Subfield-Code determines the Subfield-Code for the Target-Field's text. By default, $$a is used.

9 Description MAB-to-MARC convit-Programs

Overview

9.1	mab2usm_002	-	Processing mab-field 002 (Date)
9.2	mab2usm_037	-	Processes mab-field 037 (Language codes)
9.3	mab2usm_050	-	Processes mab-fields 050, 051, …
9.4	mab2usm_100	-	Processes mab-fields 100, 104, ... (Authors)
9.5	mab2usm_200	-	Processes mab-fields 200, 204, ... (Corporate body)
9.6	mab2usm_425	-	Processes mab-field 425 (Publication date)
9.7	mab2usm_451	-	Processes mab-fields 451, 461, ... (Common title in original form)
9.8	mab2usm_540	-	Processes mab-fields 540, 541, ... (Standard numbers)
9.9	mab2usm_902	-	Processes mab-fields 902, 907, ... (Subjects)
9.10	mab2usm_700	-	Processes mab-field 700 (Notation)
9.11	mab2usm_800	-	Processes mab-fields 800, 802, ... (secondary entries)

9.1 mab2usm_002–Processing mab-field 002 (Date)

The program processes the MAB-field 002. This information is used to generate new fields or to correct existing 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 updates and dependencies, mostly relating to position, are not 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 according to the above description. Non-matching fields are not processed.
- "#####" could be entered to the first column, but "002##" narrows the program calls considerably. For optical/documentary reasons you can also create a specific line for each relevant field so that you do not forget a single relevant field.
- Target-Field-Code, Source-Subfield-Code, and Target-Subfield-Code are not analyzed and are irrelevant.

9.2 mab2usm_037–Processing mab-field 037 (Language Codes)

The program processes the 037-MAB-fields. This information is 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 updates and dependencies, mostly relating to position, are not 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 according to the above description. Non-matching fields are not processed.
- "#####" could be entered to the first column, but "037##" narrows the program calls considerably. For optical/documentary reasons you can also create a specific line for each relevant field so that you do not forget a single relevant field.
- Target-Field-Code, Source-Subfield-Code, and Target-Subfield-Code are not analyzed and are irrelevant.

9.3 mab2usm_050–Processing mab-fields 050, 051, ...

The program processes the 050-MAB-fields. This information is used to generate new fields or to correct existing 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 updates and dependencies, mostly relating to position, are not 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 according to the above description. Non-matching fields are not processed.
- "#####" could be entered to the first column, but "050##" narrows the program calls considerably. For optical/documentary reasons you can also create a specific line for each relevant field so that you do not forget a single relevant field.
- Target-Field-Code, Source-Subfield-Code, and Target-Subfield-Code are not analyzed and are irrelevant.

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

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

The source subfield $$a is split to subfield $$a and subfield $$e. The action designator, taken from the last squared brackets, is placed in $$e. Squared brackets and possible non-sorting-characters, in front of or inside the brackets, are not adopted.

The action designator with brackets and non-sorting-characters is removed from subfield $$a.

In Source-Field-Code "**100**" the field "**1001**" is 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## and 104## to 196##. All other fields are not processed. "#####" could be entered to the first column, but "1####" narrows the program calls considerably.

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

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

The source-subfield $$a is adopted and formatted. If there is text inside angle brackets (not double-angles, which are non-sorting-characters in ALEPH), these brackets are replaced.

If Subfield $$a contains the delimiter "**/**" (blank-slash-blank), the field is split at first appearance; the text in front of the delimiter goes to $$a and the text after the delimiter goes to $$b.

In Source-Field-Code "**200**" the field "**1102**" is 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## and 204## to 296##. All other fields are not processed. "#####" could be entered in the first column, but "2####" narrows the program calls considerably.

9.6 mab2usm_425–Processing mab-field 425 (Year of Publication)

The program processes the MAB-field 425. This information is 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 not equal to "0000" in Subfield $$a is interpreted as a year of publication and is saved at a specific position in target field “008”, if this position is not occupied by a year-value.

A detailed description of the specific updates and dependencies, mostly relating to position, are not 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 according to the above description. Non-matching fields are not processed.
- "#####" could be entered in the first column, but "425##" narrows the program calls considerably. For optical/documentary reasons you can also create a specific line for each relevant field so that you do not forget a single relevant field.
- Target-Field-Code, Source-Subfield-Code, and Target-Subfield-Code are not analyzed and are irrelevant.

9.7 mab2usm_451–Processing mab-fields 451, 461, ... (Common Title in Original Form)

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

The source-subfield $$a is adopted and formatted. If Subfield $$a contains the delimiter " ; " (blank-semicolon-blank), the field is split at the 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 is not analyzed by the program.
- This is why the program has to be linked to the specific relevant fields.
- Source-Subfield-Code and Target-Subfield-Code are not analyzed and are irrelevant.

9.8 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 is adopted and formatted. If the first string without blanks (character string until first blank) contains no numeric character, this text is deleted. Normally, this is the "lead" text ISBN, ISMN, ISSN, or ISRN.

If the next string without blanks contains at least one numeric character, this string is interpreted as the standard book number. This number is saved to subfield $$a and the rest of the text to Target-Subfield $$c.

If the first string contains no numeric character, the complete content from Source-Subfield $$a is saved to Target-Subfield $$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 is not analyzed by the program (with one exception):
  - If the Target-Field-Code = "**022##**", the subfield $$c is not generated because this is not provided in the MARC-field 022. The data is lost.
- This is why the program has to be linked to the specific relevant fields.
- Source-Subfield-Code and Target-Subfield-Code are not analyzed and are irrelevant.

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

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

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

"**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 $$a is adopted as Target-Subfield $$a at the 1. field of the string. The Source-Subfield $$a of all the following fields of a string is 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 are always removed from the Source-Subfield-Text von $$a before the adoption.

- Removing of the first three 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 is not analyzed in the program.
- This is why the program has to be linked to the specific relevant fields.
- The Target-Field-Code is determined by the program.
- Source-Subfield-Code and Target-Subfield-Code are not analyzed and are irrelevant.

9.10 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 "**|**".The first 20 characters are interpreted as a link number, which should not be adopted. If the first character is the pipe "**|**", only this character is 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 section 6.6 edit_field–General Field Editing)

- The meaning of column 1 to 4 is equal to the standard described in section 5.1 Standard Processing of a Table Row.
- The processing of edit_field is continued, except from the above explanations. You may code specific parameters for that.

9.11 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 "**|**".The first 20 characters are interpreted as a link number, which should not be adopted. If the first character is the pipe "**|**", only this character is 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 section 6.6 edit_field–General Field Editing)

The meaning of column 1 to 4 is equal to the standard described in section 5.1 Standard Processing of a Table Row The processing of edit_field is continued, except from the above explanations. You may code specific parameters for that.