



System Librarian's Guide - Cataloging

Version 22

CONFIDENTIAL INFORMATION

The information herein is the property of Ex Libris Ltd. or its affiliates and any misuse or abuse will result in economic loss. DO NOT COPY UNLESS YOU HAVE BEEN GIVEN SPECIFIC WRITTEN AUTHORIZATION FROM EX LIBRIS LTD.

This document is provided for limited and restricted purposes in accordance with a binding contract with Ex Libris Ltd. or an affiliate. The information herein includes trade secrets and is confidential.

DISCLAIMER

The information in this document will be subject to periodic change and updating. Please confirm that you have the most current documentation. There are no warranties of any kind, express or implied, provided in this documentation, other than those expressly agreed upon in the applicable Ex Libris contract. This information is provided AS IS. Unless otherwise agreed, Ex Libris shall not be liable for any damages for use of this document, including, without limitation, consequential, punitive, indirect or direct damages.

Any references in this document to third-party material (including third-party Web sites) are provided for convenience only and do not in any manner serve as an endorsement of that third-party material or those Web sites. The third-party materials are not part of the materials for this Ex Libris product and Ex Libris has no liability for such materials.

TRADEMARKS

"Ex Libris," the Ex Libris bridge, Primo, Aleph, Alephino, Voyager, SFX, MetaLib, Verde, DigiTool, Preservation, URM, Voyager, ENCompass, Endeavor eZConnect, WebVoyage, Citation Server, LinkFinder and LinkFinder Plus, and other marks are trademarks or registered trademarks of Ex Libris Ltd. or its affiliates.

The absence of a name or logo in this list does not constitute a waiver of any and all intellectual property rights that Ex Libris Ltd. or its affiliates have established in any of its products, features, or service names or logos.

Trademarks of various third-party products, which may include the following, are referenced in this documentation. Ex Libris does not claim any rights in these trademarks. Use of these marks does not imply endorsement by Ex Libris of these third-party products, or endorsement by these third parties of Ex Libris products.

Oracle is a registered trademark of Oracle Corporation.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

Microsoft, the Microsoft logo, MS, MS-DOS, Microsoft PowerPoint, Visual Basic, Visual C++, Win32, Microsoft Windows, the Windows logo, Microsoft Notepad, Microsoft Windows Explorer, Microsoft Internet Explorer, and Windows NT are registered trademarks and ActiveX is a trademark of the Microsoft Corporation in the United States and/or other countries.

Unicode and the Unicode logo are registered trademarks of Unicode, Inc.

Google is a registered trademark of Google, Inc.

Copyright Ex Libris Limited, 2018. All rights reserved.

Document released: July 2018

Web address: <http://www.exlibrisgroup.com>

Table of Contents

1	RECORD FORMATS	9
2	TEMPLATES	9
2.1	Creating Local Templates	10
2.2	Creating Library-dependent Templates	10
3	VALID FIELDS	11
4	FORMS	13
5	LIST OF VALUES IN FIXED-LENGTH FIELDS FORMS	19
5.1	Defining Lists of Valid Values and Description.....	21
5.2	Load the Lists of Valid Values into Aleph Data.....	22
5.3	Setting up the "GUI Fixed-Length Fields Forms"	24
6	DEFAULT SUBFIELDS	25
7	DEFAULT FIELDS FOR NEW RECORD	26
8	TAG INFORMATION	27
9	SEARCH HEADINGS	29
10	SEARCH SUBFIELD OPTIONS	31
11	CHECK FIELD	33
11.1	AL Section	33
11.2	D section	35
12	FIX RECORD	36
12.1	tab_fix	36
12.2	fix_doc.lng	74
12.3	fix_doc_track	75
13	LOCATE FUNCTION	75

14	DUPLICATE RECORD FUNCTION.....	77
15	IMPORTING UPDATED TABLES.....	78
16	FLOATING KEYBOARD.....	78
17	AUTHORIZATIONS	82
17.1	Allowed and Denied Tags.....	82
17.2	Cataloging "OWN" Permissions.....	83
17.3	Holdings Filter	84
18	MERGING RECORDS.....	84
19	UPDATING THE TABLES PACKAGE.....	87
20	SUBFIELD PUNCTUATION	87
21	VALIDATION OF CONTENTS OF A FIELD	88
22	CHECK FIELD OCCURRENCES AND DEPENDENCY BETWEEN FIELDS	89
23	FORBIDDEN ERRORS AND TRIGGERS.....	91
24	CHECKING ROUTINES FOR NEW HEADINGS IN THE HEADINGS LIST	93
25	CHECKING ROUTINES FOR NEW HEADINGS IN THE BIBLIOGRAPHIC AND AUTHORITY HEADINGS LIST.....	93
26	CHECKING ROUTINES FOR NEW DIRECT INDEXES (IND).....	94
27	LOCKING RECORDS	95
27.1	Locking Period for Locked Records	95
27.2	Lock Status Message.....	95
28	CHECK ROUTINES FOR CHECK RECORD	95
28.1	Check Types Available for Column 1 of the check_doc Table:	96
28.2	Check Programs Available for Column 2 of the check_doc Table	96

28.3	Check Programs For Document Deletion	100
29	FIXED-LENGTH FIELDS CHECKING ROUTINES.....	101
30	VALIDATION MESSAGES (TABLE-DEPENDENT)	105
31	VALIDATION MESSAGES (SYSTEM-DRIVEN)	105
32	CATALOGING PRODUCTIVITY REPORT	106
32.1	HOL Records tab of Records Editor	106
33	COLUMN HEADINGS (PC_TAB_COL.LNG AND TAB_COL.DAT) 107	
34	DEFAULT VALUES FOR FIXED FIELDS IN NEW RECORDS.....	107
35	IMPORTING RECORDS.....	108
35.1	Remote Conversions	109
36	COMBINING DIACRITICS.....	109
37	RECORD LENGTH LIMITS	110
38	HIDDEN FIELDS.....	110
39	RECORD MANAGER.....	111
40	OVERVIEW TREE.....	111
41	SETTING UP A SCRIPT FOR THE CORRECTION OF RECORDS IN ALEPH SEQUENTIAL FORMAT	116
41.1	Generic Fix Doc Script Specification	116
41.2	Script Flow.....	116
41.3	Generic Fix Doc Operations	117
	CONCATENATE-FIELDS.....	119
41.4	Generic Fix Doc (p_file_08) Script Examples.....	124
42	CLIENT SETUP (CATALOG.INI).....	126
42.1	Catalog.ini Settings	126
42.1.1	[ConvertFile].....	126

42.1.2	[Form]	128
42.1.3	[Editor]	128
42.1.4	[ExpandTemplate]	130
42.1.5	[DuplicateRecord]	130
42.1.6	[OffLine]	131
42.1.7	[Locate]	131
42.1.8	[Scan]	131
42.1.9	[HolOwnTextDefaults]	131
42.1.10	[General]	132
42.1.11	[RecordBar]	132
42.1.12	[RecordTree]	132
42.1.13	[RfidMedia]	133
42.1.14	[LOW]	133
43	CATALOGING TABLES.....	134
43.1	Library Tables	134
44	SETTING UP THE LKR FIELD.....	138
44.1	tab_fix_z103	138
45	SUPPORTING ADDITIONAL FILTERS IN LKR FIELD	140
46	LKR UPDATING UPON ITEM ENUMERATION AND CHRONOLOGY MODIFICATION	141
47	TAB100-RELATED ENTRIES IN CATALOGING	143
48	SETUP OF ADM LIBRARIES	147
49	MATCHING RECORDS.....	147
50	SETTING UP SERVICES	150
50.1	Retrieve Catalog Records (ret-01)	150
51	CJK UNICODE CHARACTERS	151
52	PUBLISHING	151
52.1	Initial Extract Process	151
52.2	Ongoing Extract Process	153
52.3	Name Spacing in Publishing	154
53	UPLOAD BIB AND HOLDING INFORMATION FROM ALEPH TO KERIS 155	

53.1	Tables Set-Up Configuration	155
53.1.1	KERIS Z39.50 Gate Configuration	155
53.1.2	Expand and Fix Routines Setup	156
53.1.3	Manual Upload Using the Remote Menu of Cataloging Module.....	161
	The Remote menu of the Cataloging module uploads a single document to KERIS. It supports the following updates:.....	161
53.1.4	Upload New and Updated Document Manually	161
53.1.5	Upload Deleted Document Manually.....	162
53.1.6	Upload Suppressed Document Manually	163
53.1.7	Upload Repaired Document Manually.....	164
53.1.8	Bulk Upload Using Batch Service	164
53.1.9	The Batch Output Reports.....	166
54	OUF LOADER.....	167
54.1	Instructions for Running the OUF Loader	167
54.1.1	Running from the UNIX Prompt.....	167
54.1.2	Parameters Description	168
55	PREVENTING THE AUTOMATIC CREATION OF PAR RECIPROCAL LINKS BETWEEN RECORDS	169
56	GENERATING A LOCALLY ASSIGNED CALL NUMBER IN BIBLIOGRAPHIC AND ITEM RECORDS.....	170
56.1	Creating the Call Number in the BIB Record.....	170
56.2	Check Routine for Call Number Prefix	171
57	AUTOMATIC CREATION OF 6XX FIELDS	171
57.1	The Batch Service: Create Additional Subject Heading(s) from Authority (manage-46).....	171
57.2	Defining the AUT Index code for Detecting the AUT Headings - tab_bib_aut_match.....	173
57.3	Manage-46 Service Functionality - Workflow and Example	175
57.4	Match Algorithm and Translate	175
57.4.1	Match and Translate for Create 6XX Using 1XX.....	175
57.4.2	Match and Translate for Create 6XX Using 7XX.....	178
58	AUTOMATIC TRANSLATION OF BIBLIOGRAPHIC NOTE FIELDS	180
58.1	Fix Routine for Translation - fix_doc_notes.....	180
58.2	Setting Up a List of Translations - tab_fix_notes	180
58.3	Automatic Translations – Functionality and Examples	181
58.3.1	Compare Action	181
58.3.2	Replace Action.....	181

59	LINK TO RDA TOOLKIT.....	183
60	UPDATE HOL RECORD BASED ON ITEM ARRIVAL.....	183

1 Record Formats

In ALEPH, every record in the system must be assigned a Record Format. This information is kept in the FMT field which is an ALEPH-specific field. In the Cataloging module, the content of the FMT field (in other words the record format) of the record in the Catalog Editor is displayed in the Cataloging bar. The record format is consulted by various functions in the system, such as the display of online tag information, checking procedures, input forms, templates, and so on. For example, the values in the MARC 21 008 field depend on the record's format, and therefore there are separate input forms according to the format of the record (a form for BK, a form for SE, and so on). Although formats can be added, this is not recommended because of the amount of setup and upkeep this entails. Note that for more granularity in "format type", you can use the `expand_doc_type` expand program to create a TYP field with values which can be used for indexing and display in addition to or instead of the FMT.

The list of the available record formats is defined in the `formats.lng` table located in the library's `pc_tab/catalog` directory.

Following is a sample of the `formats.lng` table:

```
BK L Books
CF L Computer file
MP L Maps
MU L Music
SE L Serials
VM L Visual materials
MX L Mixed materials
```

Key to Table:

Column 1 - Code

This is the unique code by which the system identifies the format. The code must be two characters long. This code is displayed in the Cataloging bar of the Cataloging module for the record in the Catalog Editor.

Column 2 - ALPHA

ALPHA code. Must always be L.

Column 3 - Description

Enter a description for the format. This can be up to 20 characters long.

2 Templates

There are two types of cataloging templates: local templates and library-dependent templates. Local templates are only available for the station on which they are created. Library-dependent templates are available to all librarians cataloging in the library.

2.1 Creating Local Templates

To create a **local template**, perform the following steps:

Open a cataloging record to serve as the basis for your local template.

Open the Cataloging menu and choose Create Template on Local Drive.

Enter the template file name and click OK. A message appears confirming that the template has been saved. The template is now available in the list of templates, and can be chosen from the Cataloging menu using the Open Template option. This template can now be used as a basis to catalog a new record.

Note that local templates are saved on the PC in the following folder of the GUI:
../Catalog/Template.

2.2 Creating Library-dependent Templates

To create a **library-dependent** template, perform the following steps:

Add a file containing the template to the `$data_root/pc_tab/catalog` directory.

Give the file the extension `.mrc`.

Make sure that the file contains the following values in the following columns:

Column 1: Field tag code and indicators

In addition, there are three codes for the use of the system: DB and SYSID and FMT. FMT is for the code of the record format.

Column 2 - ALPHA

ALPHA code. Must always be L.

Column 3 - Subfield codes and contents

Subfield codes are prefixed by two dollar signs (\$\$). The system codes must be defined as follows:

- The value of DB is always LOCAL.
- The value of SYSID is always 0.
- The value of FMT is the code of the record format.

Following is a MARC 21 sample template for books:

```
DB      L LOCAL
SYSID   L 0
FMT     L BK
LDR     L ^^^^^nam^^22^^^^^^u^4500
008     L ^^^^^s2000^^^^^^^^^^^^^^r^^^^^000^0^eng^d
020     L $$a
040     L $$a
080     L $$a
1001    L $$a $$b $$c $$d
2401    L $$a
2451    L $$a $$b $$c $$h
24611   L $$i $$a$$b
250     L $$a $$b
260     L $$a $$b $$c
300     L $$a
440     L $$a $$n $$p $$v
500     L $$a
```

```

502   L  $$a
5050  L  $$a
650 2 L  $$a
690   L  $$a
7001  L  $$a $$b $$c $$d
7101  L  $$a $$b
7102  L  $$a $$b
7112  L  $$a $$n $$c $$d
740   L  $$a $$h

```

An alternative easy way to create a library-dependent template is to create a template locally, and then to make it available for the entire library. First create an appropriate template on the PC (local template) and then FTP the file to the `$data_root/pc_tab/catalog` directory. This ensures the proper placement of values and the functionality of the template.

Library-dependant templates are listed, together with the local templates, in the List of Templates window. Note that only those templates defined for the home library to which the cataloger is connected are available. For example, if the librarian is connected to a bibliographic library - XXX01 - the templates defined for this library are displayed in the window. If the librarian is connected to an authority library - XXX10 - templates defined for this library are listed in the List of Templates window.

3 Valid Fields

The system librarian is in charge of defining the valid tags and aliases for the Cataloging client. The list of valid tags and aliases is activated in the Cataloging module by using the hotkey F5 or by selecting the New field (choose from list) option from the Edit menu. Valid tags and aliases are defined by editing the `codes.lng` table, located in the library's `pc_tab/catalog` directory.

The "codes" table enables you to define the following three aspects:

- Defines whether the tag is displayed in the list of tags available in the Cataloging module.
- Defines whether the tag can be edited only through a form or through a form and directly on the catalog draft.
- Defines whether or not the tag can have subfields.

Following is an example from the table:

```

!1    2 3 4 5      6              7              8
!!!!-!-!-!-!-!!!!!!-!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

LDR   N Y Y L Leader          L Leader
001   Y Y Y L Control No.    L Control Number
003   Y N Y L Control No. ID L Control Number Identifier
005   Y N Y L Date and Time  L Date and Time of Last Transaction
006   Y Y Y L Linking Field  L Linking Field
007   Y Y Y L Phys.Descrip.  L Physical Description Fixed Field
008   Y Y Y L Fixed Data     L Fixed Length Data Elements
010   Y N N L LC Control No. L Library of Congress Control Number
013   Y N N L Patent Info.   L Patent Control Information

```

Key to Table:**Column 1 - Field tag**

Enter a field tag up to 5 characters long. The # character can be used as a placeholder for indicators in positions 4 and 5. For example, entering 245## includes 2451, 2452 or 24514. Fields in the "codes" table must first be defined in the "Codes and names of MARC and ALEPH fields" table (this is done by editing the tab01.lng table of the library's tab directory).

Column 2 - Display

This column defines whether the tag can be chosen from the list of tags available in the Cataloging module. Values are *Y* and *N*. If you choose *Y*, the tag is displayed in the list. If you choose *N*, the tag will not be displayed in the list but it is still possible to add manually to the catalog draft.

Column 3 - Edit

This column defines whether the tag can be edited only through an editing form or if it is possible to edit the tag both from a form and directly from the catalog draft. Values are *Y* and *N*. If you choose *Y*, you can only edit the tag through a form. If you choose *N*, you are able to edit the tag either from a form or directly in the cataloging draft.

If Column 3 is set to 'N', then the setup of Column 4 influences whether or not the form is editable. If Column 4 is set to 'N' (that is, the field can have subfields), then form editing is possible. If it is set to 'Y' (that is, the field cannot have subfields), then the form will not be available for editing.

Note that this option should be used in conjunction with the setup of the forms. If the field is a fixed-length field and the form for the field is set as a form for fixed-length fields, enter *Y*. If you want the cataloger to edit a fixed-length field from the catalog draft, then the form for the field should be defined as a form for a non fixed-length field, and this column should be set to *N*.

If a form is not available for a field, the option to edit the field through a form will not be available.

Column 4 - Subfields

This column defines whether or not the tag can have subfields. Values are *Y* and *N*. If you choose *Y*, the cataloger will not be able to add subfields to the field. If you choose *N*, the cataloger is able to add subfields to the field.

Column 5 - ALPHA of name tag

ALPHA code. Must always be L.

Column 6 - Catalog name tag

The catalog name tag (also called an alias) can be up to 16 characters long. It can be the same name tag defined for the field in the tab01.lng table, although it does not have to be. Name tags are displayed in the catalog record in addition to the (usually numeric) field tags ONLY if the DisplayTagInfo field of the PC's catalog.ini file is set to *Y*.

Column 7 - ALPHA of description

ALPHA code. Must always be L. The description is displayed in the list of valid tags available in the Cataloging module.

Column 8 - Description

Enter a description of the field, up to 38 characters long. The description is displayed in the list of valid tags available in the Cataloging module by using the hotkey F5 or choosing the New field (choose from list) option from the Edit menu.

4 Forms

You can define the forms that catalogers use to enter data for a field. The files that define forms are located in the library's \$data_root/pc_tab/catalog directory.

The file format for cataloging forms can include up to 5 characters for defining the tag and its indicators. The possible formats for the form are the following:

nnn_xx.lng is in use for undefined indicators.

nnny_xx.lng or **nnyy_xx.lng** are in use for specific indicators.

y can be used to define a specific first indicator.

yy can be used to define specific indicators.

xx is the code for the record format, for example, BK for book. Refer to Record Formats on page 9 for more information on record formats.

lng is the code for the language (for example, *eng* for English).

The **exceptions** are for forms for the MARC 21 field 007 and the MARC 21 field 006. They are in the format **007_x.lng** and **006_x.lng**, where x is the code for the material type, as defined in the files scr_007.lng and scr_006.lng (which are also located in the library's \$data_root/pc_tab/ catalog directory).

Sample Form #1

Following is a sample form for MARC 21 field 260:

```

^1^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      ^2^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^&2
      ^3^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      ^4^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

      ^5^^^^^^^^^^^^^^^^ _30_____
      ^6^^^^^^^^^^^^^^^^ _30_____
      ^7^^^^^^^^^^^^^^^^ _30_____

###
F abc

```

```

^1=Imprint (NR)
^2=Indicators
^3=First indicator (blank,2,3)
^4=Second indicator (blank)

```

^5=Place of publication (a)
 ^6=Name of publisher (b)
 ^7=Date of publication (c)

Sample Form #2

Following is a sample form for MARC 21 fixed-length field 008:

```

^1^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^2^^^^^^^^^^^^^^^^ $6$$$$                ^3^^^^^^^^^^^^^^^^ _1
^4^^^^^^^^^^^^^^^^ _4_____             ^5^^^^^^^^^^^^^^^^ _4_____
^6^^^^^^^^^^^^^^^^ _3_____             ^7^^^^^^^^^^^^^^^^ _1 _1 _1 _1
^8^^^^^^^^^^^^^^^^ _1                   ^9^^^^^^^^^^^^^^^^ _1
^10^^^^^^^^^^^^^^^^ _1 _1 _1 _1         ^11^^^^^^^^^^^^^^^^ _1
^12^^^^^^^^^^^^^^^^ _1                   ^13^^^^^^^^^^^^^^^^ _1
^14^^^^^^^^^^^^^^^^ _1                   ^15^^^^^^^^^^^^^^^^ _1
^16^^^^^^^^^^^^^^^^ _1                   ^17^^^^^^^^^^^^^^^^ _1
^18^^^^^^^^^^^^^^^^ _3_____           ^19^^^^^^^^^^^^^^^^ _1

^20^^^^^^^^^^^^^^^^ _1

###
D ^^^^^^s2003^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^000^^^eng^d
V
  
```

- ^1=008 Fixed length data elements (BOOKS)
- ^2=Date entered on file (00-05)
- ^3=Type of date (06)
- ^4=Date 1 (07-10)
- ^5=Date 2 (11-14)
- ^6=Publication Place (15-17)
- ^7=Illustration codes (18-21)
- ^8=Target audience (22)
- ^9=Form of item (23)
- ^10=Nature of contents (24-27)
- ^11=Govt. publication (28)
- ^12=Conference publ. (29)
- ^13=Festschrift (30)
- ^14=Index (31)
- ^15=Unspecified (32)
- ^16=Literary form (33)
- ^17=Biography (34)
- ^18=Language (35-37)
- ^19=Modified record (38)
- ^20=Cataloging source (39)

Key to Form

In general terms, the files for the forms are divided into three main sections:

The upper section of the file. This section is used to define the display of the various elements of the form and to define the actual length for the input fields. The following example is from the form for the 260 field:

```

^1^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^2^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^&2
^3^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  
```

```

^4^^^^^^^^^^^^^^^^^^^^
^5^^^^^^^^^^^^^^^^ _30_____
^6^^^^^^^^^^^^^^^^ _30_____
^7^^^^^^^^^^^^^^^^ _30_____

```

###

The additional values section. The following example is from the form for the 260 field:

F abc

The lower section of the file. This section is used to define the actual text that is displayed. The following example is from the form for the 260 field:

```

^1=Imprint (NR)
^2=Indicators
^3=First indicator (blank,2,3)
^4=Second indicator (blank)
^5=Place of publication (a)
^6=Name of publisher (b)
^7=Date of publication (c)

```

Visual Text Definitions (Upper and Lower Sections)

Each line in the lower section of the file should have a matching line in the upper section. In both sections, each line must begin with the caret (^) sign followed by a number. The match between the lines is performed according to this number as shown in the examples below. Note that in the lower section, an equal sign (=) is added after the matching number. To summarize, the lower section of the forms determines the text to be displayed in the position determined by the matching line in the upper section.

Based on the form for the 260 field, the text *Imprint (NR)* is displayed at the top of the form. The following line from the lower section:

```

^1=Imprint (NR)

```

matches the following line in the upper section of the file:

```

^1^^^^^^^^^^^^^^^^^^^^

```

The text *Indicators* is displayed below *Imprint (NR)*. The line:

```

^2=Indicators

```

matches the following line in the upper section:

```

^2^^^^^^^^^^^^^^^^^^^^&2

```

The upper section not only determines the position of the text (in other words, of the lines in the lower section); it is also used to determine the length of the display text. This is done by the caret signs that follow the matching number. For this reason, when creating a new form, it is important to include enough caret signs in the line of the upper section, or the text will be cut when displayed in the Cataloging module. Note

that the caret signs are not in a relationship of one-to-one with the characters in the text line of the lower section. This occurs especially with proportional fonts, where the character width varies and the characters do not occupy the same amount of horizontal space. For example, the first line in the upper section of our sample file contains two caret signs after the matching number:

```
^1^^
```

Since the width of each letter is proportional to its shape and an *i* is narrower than a *w* in the Cataloging module, the two caret signs display up to six *w*'s and up to twenty-four *i*'s for the Tahoma font. For Courier New fixed font, up to five *w*'s and five *i*'s are displayed.

When creating new forms, it is best to proceed on a trial-and-error basis. The font for the text displayed in the form is defined in the font.ini file of the ALEPHCOM/TAB directory under the WindowControls section. Note that this section also determines the font for many other instances in the client such as buttons, edit fields, static fields and more.

Input Fields Definitions (Upper Section)

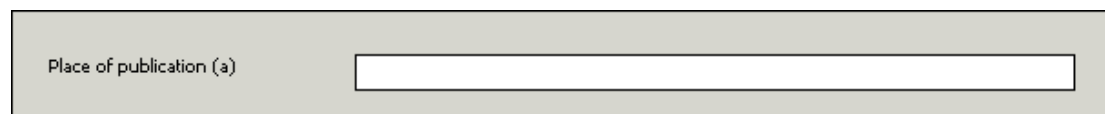
In addition to visual text definitions, the input fields for the forms are defined through the upper section of the file. The following line from the form for the 260 field, includes the text defined in line 5 of the lower section and the input field definitions:

```
^5^^^^^^^^^^^^ _30_____
```

The lower section of the file contains the following line:

```
^5=Place of publication (a)
```

This is displayed as follows in the Cataloging module:



The screenshot shows a rectangular form area with a light gray background. On the left side, the text "Place of publication (a)" is displayed. To the right of this text is a white rectangular input field with a thin black border, which is currently empty.

The input field is achieved by the following line:

```
_30_____
```

The line must begin with an underscore (`_`) character. A number that determines the actual length assigned to the subfield (in this case, to subfield \$a of the 260 field) must follow the underscore (in this example, up to 30 characters can be entered in the input field for subfield \$\$a). After the number - as with the caret sign for text - it is necessary to define the visual definitions for the input field. This is done by adding underscore characters that represent the visual display of the field. Note that the visual length includes the positions defined by the first underscore and by the following number. For example, if the line is defined as follows:

```
_30
```


then the input field is displayed in the Cataloging module as follows:

The cataloger can still type up to 30 characters but only some of the characters will be displayed (the number of characters displayed is relative to the font).

The relationship between the characters displayed in the input field and the number of underscores is not one-to-one (as with the caret sign and the characters for the text). Again, when creating new forms, it is best to proceed on a trial-and-error basis. For example, if the line is defined as follows and the font used is *Tahoma*:

`_30`

then in the Cataloging module, only up to two *w*'s will be displayed at a time, but up to four *i*'s can be displayed at a time:

The font for the input fields of the form is defined in the `font.ini` file of the `ALEPHCOM/TAB` directory under the **WindowControls** section. Note that this section also determines the font for many other instances in the client such as the **Browse** and **Find** edit fields.

Upper Section - Additional Information

Indicators: For fields that have indicators, the line for indicators must appear before the lines for text input. The ampersand (&) sign and the number 2 are used to indicate the two positions for the indicators. The following example is from the form for the 260 field:

`^2^^^^^^^^^^^^^^^^^^^^^^&2`

Dollar (\$) sign: Indicates that the information is displayed on the screen, but the user cannot change it. The following example is from the form for the 008 field:

`^2^^^^^^^^^^ $6$$$$`

This line is used for positions 00 to 05 of the 008 field and contains the date entered on file. This information is added automatically by the `fix_doc_tag_008_open_date` `fix` program. Before applying the `fix`, this input field is displayed as follows in the form:

Following is the display of this input field in the form after the `fix` program inserted the date:

Date entered on file (00-05)	030409
------------------------------	--------

Similar to the specifications for a standard field, the first character is a dollar sign (instead of an underscore). Following this first dollar sign, the line includes the actual length of the positions that are in display-mode only - in this case 6 (positions 00 to 05). After the number, it is necessary to define the visual definitions for the input field. This is done by adding additional dollar signs that represent the visual display of the field. Note that in order for the dollar signs to work, an underscore has to exist in the form for another character in another position. In other words, the form must include a character with an underscore in order for the dollar signs to work.

###: Three hash signs are used to specify the end of the upper section of the field.

Additional Values Section

The additional values section divides the upper and lower sections of the file. The following example is from the form for the 260 field:

F abc

The following example is from the form for the 008 field:

D ^^^^^^s2003^^000^^^eng^d
V

Following are the available options for this section:

F - The *F* line is used for non-fixed-length fields to indicate which subfields are defined in the form. For example, the following line:

F abc

specifies that the form includes subfields \$a, \$b and \$c. Make sure that you leave a space between the *F* and the beginning of the first subfield. Do not leave a space between subfields.

D - The *D* line is used for fixed-length fields to indicate the length of the field and the default values for the form. For example, in the following line:

D ^^^^^^s2003^^000^^^eng^d

the form for the 008 field is set to 40 character positions and, as an example, the default language in the form (positions 35 to 37) is set to "eng" (English).

Note that the whole field length needs to be specified. For this reason, positions that are set as undefined, must be marked by using the character that is used to denote a blank in fixed-length fields. In the above example, the caret (^) is used. This is the

standard for MARC 21 libraries. The following example is from the 100 UNIMARC field:

```
D -----d-----km-y0enga0103----ba
```

In the above example, the character used to denote blanks is the hyphen (-). The character used is defined in the DOC-BLANK-CHAR variable in the tab100 table of the library's tab directory.

V - The *V* indicates that the form should be verified for correctness before letting the user leave the form. If errors were found, the error messages are shown in the Messages tab of the lower pane and the system displays a prompt informing the cataloger that checking the field reported warnings and asking if he still wants to continue with the closure of the form.

S - The *S* line is used for fixed-length fields that have subfields (for example, the 100 UNIMARC field). This type of line is used to define the subfields for the form and the length of each subfield. The following is a sample line from the form for the 100 UNIMARC field:

```
S a(36)
```

The above line indicates that the form contains subfield \$a of size 36. The following is an additional example:

```
S a(11)b(7)c(12)d(9)
```

The above line indicates that the form contains subfields \$a, \$b, \$c, and \$d. Subfield \$a is of size 11, subfield \$b is of size 7 and so on.

5 List of Values in Fixed-Length Fields Forms

Depending on your system set-up, the cataloging forms of fixed-length fields can contain an option to display a list of valid values for certain positions.

Each library decides for which forms and for which positions the Valid Value dialog box should be available.

Below is a sample of the MARC 008 Fixed Length Form with an expand button next to the Place of Publication Code field (008 positions 15-17):

MARC 008 Fixed length data elements (CONTINUING RESOURCES)

Date entered on file (00-05)	871006	Type of date (06)	c
Date 1 (07-10)	1898	Date 2 (11-14)	9999
Place of publication code (15-17)	be ...	Frequency (18)	a
Regularity (19)	r	ISSN center (20)	
Type of continuing resource (21)		Form of original item (22)	
Form of item (23)		Nature of entire work (24)	
Nature of contents (25-27)		Govt. publication (28)	
Conference publ. (29)	0	Undefined (30-32)	
Alphabet/Script (33)		Entry convention (34)	0
Language (35-37)	fre ...	Modified record (38)	1
Cataloging source (39)	d		

OK
Cancel

Clicking the expand button displays the List of Valid Values dialog box, which contains a list of pre-defined values and descriptions for certain tags and positions.

Below is a sample of List of Values for tag 008 positions 15-17: Place of Publication:

List of Values

Sort by...

Value
 Description

Enter Starting Point
ja

Value	Description
ja	Japan
jm	Jamaica
jo	Jordan
ke	Kenya
kn	Korea
kz	Kazakhstan
lh	Liechtenstein
lu	Luxembourg
lv	Latvia
mau	Massachusetts
mm	Malta

Select

Cancel
Help

The content of the List of Values is determined by the library's setup. The staff user can navigate the list and select an entry to be populated in the relevant position of the fixed length form.

Perform the following in order to configure the display of the List of Values dialog box for fixed-length fields.

- 1) Define lists of valid values and descriptions.
- 2) Load the lists of valid values into Aleph data.
- 3) Set up the "GUI Fixed-Length Fields Forms" to call up the relevant list.

5.1 Defining Lists of Valid Values and Description

The list of valid values and their description is defined in configuration tables.

The table file names are determined by the library. The language extension <lng> must be part of the file name. There should be separate tables for each language.

The Exlibris sample demo table is located in ./usm01/tab directory and is named: tag_values.eng.

Table structure:

Col.1 – Identifier for the list (upper case string, up to 30 characters). The identifier represents a list of values and description for certain material type/tag/position. This identifier should be set in the 'Fixed-Length Form' file. Each field in the 'Fixed-Length Form' that calls the specific identifier lists all Values and Descriptions of the quoted identifier. Each library determines its own identifier codes.

Col. 2 – Value that is valid for the identifier (up to 10 characters)

Col.3 – Description of the value (up to 50 characters)

The following are a few samples of possible identifiers:

Identifier: PLACE-OF-PUBLICATION – Used for setting values and descriptions for tag 008 positions 15-17 (all material types).

Identifier: ILLUSTRATION-CODE-BK – Used for setting values and descriptions for tag 008 positions 18-21 (material type: BK).

Identifier: FORM-OF-COMPOSITION-MU – Used for setting values and descriptions for tag 008 positions 18-19 (material type: MU).

Below is a partial sample from Exlibris demo tables ./usm01/tab/tag_values.eng. This sample defines valid values and descriptions for the above identifiers.

! 1	2	3
!!-!!!!!!!!!!!!-!!->		
PLACE-OF-PUBLICATION	aa	Albania
PLACE-OF-PUBLICATION	abc	Alberta
PLACE-OF-PUBLICATION	-ac	Ashmore and Cartier
PLACE-OF-PUBLICATION	ae	Algeria
!		
ILLUSTRATION-CODE-BK	a	Illustrations
ILLUSTRATION-CODE-BK	b	Maps
ILLUSTRATION-CODE-BK	c	Portraits
ILLUSTRATION-CODE-BK	d	Charts

!		
FORM-OF-COMPOSITION-MU	an	Anthems
FORM-OF-COMPOSITION-MU	bd	Ballads
FORM-OF-COMPOSITION-MU	bg	Bluegrass music
FORM-OF-COMPOSITION-MU	bl	Blues

The content of the tag value tables must be loaded into Aleph data (Z112 Oracle table). For details, see the Load the lists of valid values into Aleph data section.

NOTES:

- The configuration table file names can be decided per site. The demo suggested file name (tag_values.eng) is not mandatory. A library may name the files differently. The language extension <.lng> must be part of the file name.
- A library may decide either to have a single table for all valid values and descriptions (e.g. file name: ./xxx01/tab/tag_values.eng), to have a set of tables per tag (e.g.: 006_values.eng, 007_values.eng and 008_values.eng), or to have tables per tag and format (e.g.: 006_se_ values.eng, 006_bk_valuers.eng, 008_se_values.eng, etc).
- Different MARC formats (like MARC21 and KORMARC) hold their set of tables at the relevant BIB library so that different values can be set to different formats.
- The valid values and the description may contain characters with diacritics, such as the Ä, É, Ö, Ü letters (with umlaut or accent mark) in the words: Ägypten, Égypte, Österreich, and Türkiye. In order to sort these letters in the right order, for example, "Ä" after "A" and not after "Z", add in
 ./alephe/unicode/tab_character_conversion_line a line with type "SORT-z112", using a table in which the diacritics have fallback characters without diacritics. For example:

SORT-Z112	##### # line_utf2line_utf	unicode_to_word_gen
-----------	---------------------------	---------------------

5.2 Load the Lists of Valid Values into Aleph Data

Once the identifiers are set in the valid values configuration tables (e.g. ./usm01/tab/tag_values.eng), their content should be loaded into the BIB XXX01 Aleph Oracle table: Z112 (Fixed Length Tag Values table). The load is performed by using the batch service: "Import Database Tables - With Checks (file-06)".

Import Database Tables - With Checks (file-06) - USM01

*Input File: tag_values.eng

*Oracle Table: Z112

Procedure to Run: Add New Record

Special Fix Procedure to Run: No Special Fix Routine

Correct Record: No

Check Record: No

Character Conversion: None

Runtime: Today

At: O'clock:

Library: USM01

Buttons: Submit, View History, Cancel, Help, Add to History (checked), Print To ADM Lib (unchecked)

Input file:

The file-06 service requires that the input file is located in the library's DATA_FILES directory (e.g.: /xxx01/files). This means that the valid value tables should be copied to DATA_FILES before submitting file-06. Alternatively, the user may place the exact path from DATA_FILES to the file location (e.g.: ../tab/tag_values.eng) in the "Input File" field.

Oracle table:

Should be : "Z112".

Procedure to Run

Must be: "Add New Record".

Special Fix Procedure to Run:

Should be: "No Special Fix routine".

Correct Record and Check Record

Should be: "No".

Character Conversion

For a non-Latin input file (e.g.: tag_values.kor), select the required Character Conversion routine to be applied on the imported Z112 table.

Note: Each time a change is made in the valid values configuration table/s, it is recommended to drop Z112 data (using util A/17/1) and re-load the content of the configuration table/s.

5.3 Setting up the "GUI Fixed-Length Fields Forms"

Once the list of identifiers is set in the valid values tables, and the data is loaded into Z112 Oracle table, the GUI "Fixed Length Forms" should be amended to call the relevant identifiers. This causes the relevant list to be called-up when a "Fixed-length Form" is opened in the Cataloging module.

For this purpose, each of the forms that should contain list of valid values and descriptions must be adjusted to contain the relevant identifier. The cataloging form directory is: ./xxx01/pc_tab/catalog.

For example:

To enable the values and description list in 008 form of BK format, the file ./xxx01/pc_tab/catalog/008_bk.eng must be set with relevant identifiers. The identifier should be set next to the fields. Two percent sign (%%) should precede the identifier code as in the sample below of 008_bk.eng (the add-on elements are in **bold text**).

```

^1^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^2^^^^^^^^^^^^^^^^ $6$$$$                ^3^^^^^^^^^^^^^^^^ _1
^4^^^^^^^^^^^^^^^^ _4___                  ^5^^^^^^^^^^^^^^^^ _4___
^6^^^^^^^^^^^^^^^^ _3___                  ^7^^^^^^^^^^^^^^^^ _1 _1 _1 _1
^8^^^^^^^^^^^^^^^^ _1                     ^9^^^^^^^^^^^^^^^^ _1
^10^^^^^^^^^^^^^^^^ _1 _1 _1 _1           ^11^^^^^^^^^^^^^^^^ _1
^12^^^^^^^^^^^^^^^^ _1                    ^13^^^^^^^^^^^^^^^^ _1
^14^^^^^^^^^^^^^^^^ _1                    ^15^^^^^^^^^^^^^^^^ _1
^16^^^^^^^^^^^^^^^^ _1                    ^17^^^^^^^^^^^^^^^^ _1
^18^^^^^^^^^^^^^^^^ _3___                 ^19^^^^^^^^^^^^^^^^ _1
^20^^^^^^^^^^^^^^^^ _1

D ^^^^^^s2003^^^^^^^^^^^^^^^^^^^^^^^^^^^^000^^^eng^d
V

^1=008 Fixed length data elements (BOOKS)
^2=Date entered on file (00-05)
^3=Type of date (06)
^4=Date 1 (07-10)
^5=Date 2 (11-14)
^6=Publication Place (15-17)%%PLACE-OF-PUBLICATION
^7=Illustration codes (18-21)%%ILLUSTRATION-CODE-BK

```



```

^8=Target audience (22)
^9=Form of item (23)
^10=Nature of contents (24-27)
^11=Govt.publication (28)
^12=Conference publ. (29)
^13=Festschrift (30)
^14=Index (31)
^15=Unspecified (32)
^16=Literary form (33)
^17=Biography (34)
^18=Language (35-37)
^19=Modified record (38)
^20=Cataloging source (39)

```

The result of the above implementation is that the GUI Fixed-length form for format BK tag 008 has an expand button next to the fields: "Publication Place" (position 15-17) and "Illustration Code" (positions 18-21). When the expand button is clicked, the relevant list of values is displayed.

Note: After updating the cataloging forms in `./xxx01/pc_tab/catalog`, run util M/7 utility to update the PC forms package.

6 Default Subfields

You can define the default subfields for fields by editing the `marc_exp.dat` table located in the library's `pc_tab/catalog` directory. The subfields defined in this table are displayed in the following circumstances:

When a field is selected from the list of valid fields - available by using the F5 shortcut key or by selecting the New field (choose from list) option from the Edit menu.

When the Open form option from the Edit menu is chosen for a field for which no form is available.

Note that you do not have to define all subfields, just the most common ones. This is because the cataloger can manually add to the catalog record subfields that are not included in the list of defaults.

Following is an example from the table:

```

1  2  3      4
!!!-!!!-!!!-!!!!!!
011   XX a
017   XX ab
024 #  MU adz
025   XX a
027   XX a
028 ## MU ab

```

032 SE ab
033 ## XX abc

Key to Table:

Column 1 - Field tag

Enter a 3-character field tag.

Column 2 - Indicators

Enter specific indicators, or use the "#" character as a wildcard to indicate any indicator.

Column 3 - Record Format

Enter a specific record format, or use *XX* as a wildcard to indicate that the field is appropriate for any format. Refer to Record Formats on page 9 for more information on record formats.

Column 4 - Subfields

Enter the subfields that you want to be displayed as defaults. Enter them one after the other, without spaces between them. For example: acd.

Remember that you do not have to define ALL subfields, just the most common ones. This is because the cataloger can manually add subfields to the catalog record that are not included in the list of defaults.

7 Default Fields for New Record

You can define the fields that will appear automatically in a new record when the cataloger chooses New Record from the Cataloging menu.

Default fields for new records are defined in the `tagonnew.dat` table located in the library's `pc_tab/catalog` directory.

Following is an example from the table:

```
!!!!!!  
L008  
L007
```

Enter each desired field on a separate line, in the order in which you want the fields to appear.

If the field has a form, then the form will open automatically when the cataloger creates the new record. If no form is available, the tag without indicators or subfields is displayed ready for the cataloger to edit.

Note that the LDR field is inserted automatically when a new record is created, there is no need to define it in the `tagonnew.dat` table.

8 Tag Information

The Tag Information tab (lower pane of the Cataloging tab) provides a guide to the use of valid indicators, subfield codes and values for the field selected in the Catalog Editor (upper pane). The help displayed in the Tag Information tab is defined in a separate HTML file for each field. These help files are located in the library's /pc_tab/catalog/html directory.

The following are the naming conventions for these files:

nnn_xx_lng.html is in use for undefined indicators (for example, 100_xx_eng.html)

nnny_xx_lng.html or **nnnyy_xx_lng.html** are in use for specific indicators (for example, 853x_xx_eng.html)

y can be used to define a specific first indicator.

yy can be used to define specific indicators.

xx is the code for the record format, for example, BK for book. Refer to Record Formats on page 9 for more information on record formats. The notation *xx* is used to specify that the help is accurate for any record type.

lng is the code for the language (for example, *eng* for English).

The following is the help file for the 260 MARC 21 field (260_xx_eng.html):

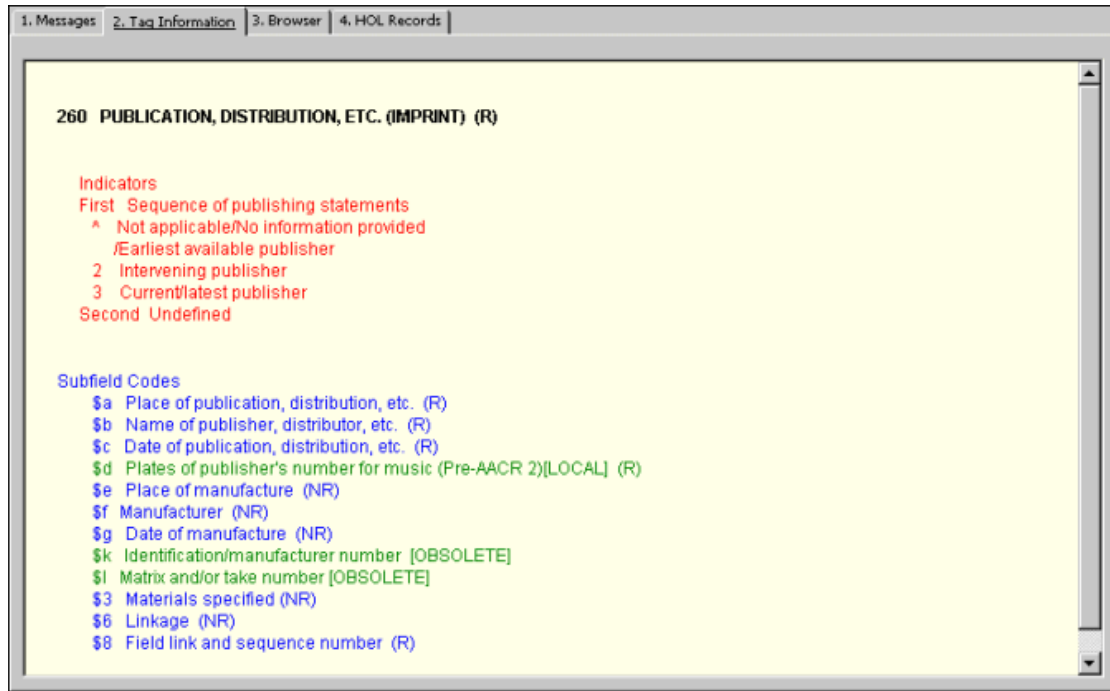
```
<html>
<head>
<include>style
</head>
<body>
<pre>
<div class=ct1>
    260 PUBLICATION, DISTRIBUTION, ETC. (IMPRINT) (R)</div>
<div class=ct2>
    Indicators
    First Sequence of publishing statements
        ^ Not applicable/No information provided
          /Earliest available publisher
        2 Intervening publisher
        3 Current/latest publisher
    Second Undefined
</div>
<div class=ct3>
    Subfield Codes
        $a Place of publication, distribution, etc. (R)
        $b Name of publisher, distributor, etc. (R)
        $c Date of publication, distribution, etc. (R)
        <span class=ct4>$d Plates of publisher's number for
music (Pre-AAC
R 2)[LOCAL] (R)</span>
        $e Place of manufacture (NR)
        $f Manufacturer (NR)
        $g Date of manufacture (NR)
        <span class=ct4>$k Identification/manufacturer number
[OBSOLETE]
        $l Matrix and/or take number [OBSOLETE]</span>
        $3 Materials specified (NR)
        $6 Linkage (NR)
```

```

                $8    Field link and sequence number    (R)
</div>
</pre>
</body>
</html>

```

This is displayed as follows in the Cataloging module:



The different display elements - such as background color, text color and fonts - are defined in a file named *style* located in the same directory. Following is a sample of a section from the style file:

```

.ct1
{
  background-color:#FFFFFFE7;
  font-size:12;
  font-weight:bold;
  font-family:Arial, Helvetica, serif;
  color:#000000;
  text-decoration:none;
}

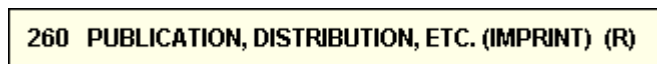
```

The above section controls the display of the title from the 260 example:

```

<div class=ct1>
  260    PUBLICATION, DISTRIBUTION, ETC. (IMPRINT)    (R)</div>

```



9 Search Headings

You can define the headings file that should be used when the cataloger chooses one of the Search Headings functions by editing the `scancode.dat` table located in the library's `pc_tab/catalog` directory. The following is a sample of the table:

```
! 1 2 3 4
5
!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!->
LOCAL USM01 USM10
USM12
050## LCC
086## SUD
100## d AUT PER
110## AUT COR
111## AUT MET
130## TIT TIT
245## TIT
260##a PLA
260##b PUBKey to Table:
```

Key to table

Column 1 - Local

The Local column lists the field tags, indicators, and subfields for which the cataloger can search a headings list. The # character can be used as a placeholder for indicators in positions 4 and 5.

For example:

`245##` includes `2451`, `2452`, `24515`, and so on.

When using specific indicators make sure that specific lines are listed before general lines, since the first match found is always taken. For example, `24510` must be listed before `245##`. If the table is defined in the following order:

```
245##
24510
```

then the `24510` field in the record will match the first line (`245##`), and not the second line.

If the lines are not sorted, and therefore `245##` is listed first, the search looks for matches to `245##`.

You can define subfields for the cataloger to search by placing the subfield code next to the field tag and indicator (see MARC 21 field `260` in the above sample table).

Column 2 - Optional.

You can specify up to 20 subfields that are not to be overwritten. When a heading is selected from a list of headings displayed by any of the Search Headings options, the selected heading replaces the entire field (therefore, all subfields) in the record unless specified in this column.

Columns 3-11

At the top of each column is the code of a searchable base. Below that, for each field tag, is the code of the headings list that is displayed for the base. The code must be a valid headings index code.

For the Search Field Headings of other Base and the Search Subfield Headings of other Base options, it is possible to specify a base more than once in the `scancode.dat` table to enable the user to define more than one scan code for the field or subfield option. For example, if the `scancode.dat` table is defined as follows:

```

! 1      2                      3                      4
5
                                6
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!->
LOCAL          USM01                      USM10
USM01
                USM01
100##  d      AUT                      PER
AUT
                SUB

```

When selecting the Search Field Headings of Other Base option for the 100 field, the cataloger is prompted to select one of the following options:

- Scan USM10 with PER
- Scan USM01 with AUT
- Scan USM01 with SUB

Note that the first base specified in the table is the local library/base and it is not displayed when selecting to scan the headings from other bases. For this reason, to enable the user to select this option when using this function, the column must be repeated.

In order to be able to search the base in the first line (Local), it has to be defined in the `searbase` file which resides in `alephcom\tab`.

Note that the maximum number of lines that the table can contain is 200.

In addition, note that the variable `USE-ACC-TEXT` in the `tab100` table of the Authority library is used to define whether the preferred heading from the authority record is taken when selecting a heading that is a "See From" reference from the list of authority headings displayed when the "Search field/subfield headings of other base" options are used. If the variable is left blank or is set to N, then when selecting a "See From" heading from the "Search field/subfield headings of other base" (Ctrl + F3/F4) options, the preferred form of the heading is inserted automatically into the catalog draft. If the flag is set to Y, the selected heading (even if it is the non-preferred form of the heading) is inserted into the cataloging draft.

When the "Search field/subfield headings of current base" options are used, if the heading is not connected to an authority record, then the heading is taken as is and inserted into the cataloging draft.

If the order of the subfields in the cataloging draft is different than the order of the heading subfield, the subfield order in the cataloging draft remains and the additional subfields are appended at the end of the cataloging draft record.

Note that this situation can result in a new index entry where the subfield content is the same as a pre-existing index entry, but the subfields are ordered differently. In such a situation, you may need to update the cataloging draft.

If the heading is connected to an authority record, the system checks the USE-ACC-TEXT variable:

If the flag is set to N (or is left blank), the system automatically inserts the preferred form of the heading from the associated authority record.

If the flag is set to Y and the heading is a "See From" heading pointing to another heading (to the preferred form of the heading) in the bibliographic list of headings, then the system takes the heading to which the "See From" is pointing (from the bibliographic database and not from the authority record).

If the flag is set to Y and the heading does not point to another heading in the database (in other words, the heading is connected to an authority record but it is the preferred heading), then the system takes the heading as is from the bibliographic database and not from the authority record.

When you run one of the Search Headings functions, the text taken from the cataloging draft undergoes non-filing procedures to strip non-filing text before the search is performed. The stripping is performed according to the non-filing indicator specified in the tab01.lng table of the library's tab directory.

10 Search Subfield Options

The tag_text.dat table located in the library's pc_tab/catalog directory is used to define the menu options that are displayed when the cataloger chooses Search Subfield Options. Following is a sample of the table:

```
!1  2  3  4      5
!!!-!!-!-!-!!!!!!-!->
655 ## L a Biographies
655 ## L a Bird's eye views
655 ## L a Cartoons
655 ## L a Catechisms
655 ## L a Daybooks
655 ## L a Diaries
655 ## L a Directories
655 ## L a Essays
655 ## L a Hymns
655 ## L a Journals
655 ## L a Memoranda
655 ## L a Questionnaires
655 ## L a Reviews
655 ## L a Syllabi
655 ## L a Time sheets
```

LKR ## L a UP//Up link
 LKR ## L a DN//Down link
 LKR ## L a ANA//Analytical link
 LKR ## L a PAR//Parallel link
 LKR ## L a ITM//Item link

Key to Table:

Column 1 - Field tag

Enter a 3-character field tag.

Column 2 - Indicators

You can enter a specific indicator, or use the # character as a wildcard to indicate any indicator.

Column 3 - ALPHA

ALPHA code. Must always be L.

Column 4 - Subfield

Enter the subfield for which you are providing a menu option. You can use the # character as a wildcard to indicate any subfield.

Column 5 - Text for menu option

Enter the text as you want it to appear in the Choose Subfield Text window displayed when the Search Subfield Options function is used. You can enter up to 45 characters. You can use two slashes // to separate the actual value from additional text that appears in the menu but will not be entered in the catalog record. See the LKR field in the above section, where "UP" is the value that is entered in the catalog record, and "Up link" is additional text that is displayed in the menu.

In addition to defining menu options, you can have the system check the validity of text entered into subfields when the user chooses the Check Record function, accessible from the Edit menu. To do so, follow these steps:

Edit the check_doc_tag_text table located in the library's tab directory. This table lists valid text for each field that you want to be checked.

The structure of the table is similar to the table used to define the text options; the following is a sample of the table:

```
! 1   2 3                               4
!!!!-!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
655## L a Biographies
655## L a Catechisms
655## L a Essays
655## L a Hymns
655## L a Reviews
655## L a Daybooks
655## L a Diaries
655## L a Directories
655## L a Journals
655## L a Memoranda
655## L a Questionnaires
655## L a Syllabi
655## L a Time sheets
655## L a Bird's eye views
655## L a Cartoons
```


Note that this table contains the valid values to be checked and should not include the description that can be added to the `tag_text.dat` table.

The `check_doc_tag_text` table can be used independently from the `tag_text.dat` table to check text validity for subfields without enabling the Search Subfield Options.

Ensure that the "check_doc_tag_text" program is listed in the `check_doc` table of the library's `tab` directory. The `check_doc` table lists all available checking programs.

11 Check Field

You can define the checks that are made when the cataloger chooses the Check Field function. To define them, edit the `check_doc_line` table located in the library's `tab` directory. There are two sections in this table:

- AL
- D

11.1 AL Section

This section enables you to define the following checks:

- Valid indicators and/or subfield codes for the tag.
- Presence of mandatory subfields.
- Non-repeatability of non-repeatable subfields.

Following is a sample of the section:

```
!!-!!-!!!!-!!!!!!-!-!-!  
AL XX      260  -  
AL XX      260  a 0 -  
AL XX      260  b 0 -  
AL XX      260  c 0 -  
AL XX      260  d 0 -  
AL XX      260  e 0 1  
AL XX      260  f 0 1  
AL XX      260  g 0 1  
AL XX      260  6 0 1
```

Key to the AL Section

Column 1 - Section ID

Enter *AL* for each line in this section of the table.

Column 2 - Record Format

Enter a specific record format, or use *XX* as a wildcard to indicate that the field is appropriate for any format. Refer to Record Formats on page 9 for more information on record formats.

Column 3

Not in use.

Column 4 - Field tag

Enter a field tag.

Column 5 - Subfield/Indicators

Enter the subfield that you want to be included in the check.

To define valid indicators for the tag, enter a hyphen (-) in this column.

Note that the indicator portion (for all formats) must be listed before the subfield portion, for each field.

Column 6 - Mandatory - Non-mandatory subfield / Valid first indicator

If column 5 contains a subfield code, this column is used to define whether the subfield is a mandatory subfield of the field. Values are 0 and 1. If the subfield is mandatory, enter 1. If the subfield is optional, enter 0.

If column 5 contains a hyphen, this column is used to define possible values for the first indicator of the field.

Column 7 - Repeatable subfield / Valid second indicator

If column 5 contains a subfield code, this column is used to define the repeatability of the subfield. Values are 1 - 9 and hyphen (-). If the subfield is not repeatable, enter 1. If the subfield can be repeated unlimitedly, enter hyphen (-). You can use values 2 - 9 to determine that the subfield can be repeated up to the number of times represented by the selected value.

If column 5 contains a hyphen, this column is used to define possible values for the second indicator of the field.

Example:

AL XX	020	-	
AL XX	020	a	0 1
AL XX	020	c	0 1
AL XX	020	z	0 -
AL XX	020	6	0 1
AL XX	022	-	
AL XX	022	-	0
AL XX	022	-	1
AL XX	022	a	0 1
AL XX	022	y	0 -
AL XX	022	z	0 9
AL XX	022	6	0 1

In the above example:

For tag 020:

- Both first indicator and second indicator are undefined (blank)
- All subfields \$a, \$c, \$z, and \$6 are optional
- Subfield \$a, \$c and \$6 are not repeatable

- Subfield \$z is repeatable an unlimited number of times

For tag 022:

- First indicator can be blank, 0 or 1
- Second indicator is blank
- All subfields \$a, \$y, \$z, and \$6 are optional
- Subfields \$a and \$6 are not repeatable
- Subfield \$y is repeatable an unlimited number of times
- Subfield \$z can be repeated up to 9 times

11.2 D section

This section enables you to determine the rules for checking dependencies among subfields of a single field. Following is a sample of the section:

```

1  2  3    4    5 6              7 8      9      1011      12
13
!!-!!-!!!!-!!!---!-!!!!!!!!!!!!!!!!!!!!-!-!-!!!!!!!!!!!!!!!!!!!!-!-!-
!!!!!!!!!!!!!!!!!!!!-!-

D  XX 9036 260   a              Y b              Y c
N
D  XX 9036 300   a              Y c              N
D  XX 9036 300   a              Y                b
N

```

Key to D section of table:

Column 1 - Section ID

Enter *D* for each line in this section of the table.

Column 2 - Record Format

Enter a specific record format, or use *XX* as a wildcard to indicate that the field is appropriate for any format. Refer to Record Formats on page 9 for more information on record formats.

Column 3 - ID # of error message

Choose the ID # of the error message that is appropriate for the check. The message is displayed to the cataloger when the system performs a check and finds an error.

User-defined messages can be defined in the library's `check_doc.lng` table located in the library's `tab` directory.

Column 4 - Field tag

Enter a field tag.

Column 5 - Subfield

Enter the subfield that you want to be included in the check. See also column 8 below.

Column 6 - Subfield contents

Optional. Enter any contents that you want the system to check for.

Column 7 - Type of dependency

Enter *Y* if the subfield should be present. Enter *N* if the subfield must not be present.

Columns 8, 9, 10 and 11, 12, 13

The subfield, contents, and dependency columns are repeated. This enables you to create *if, then* statements. For example, you can say that if subfield \$a appears (defined in columns 5, 6, 7), then subfield \$b must appear (defined in columns 8, 9, 10), and subfield \$c must not appear (defined in columns 11, 12, 13).

12 Fix Record

Fix routines are standard library-defined procedures that automatically "fix" or make changes to cataloging records. The system librarian is in charge of defining which fix programs are available and when they are run. Two tables are involved in the setup of fix procedures:

`tab_fix` (located in the library's `tab` directory)

`fix_doc` (located in the library's `pc_tab/catalog` directory)

12.1 `tab_fix`

The `tab_fix` table defines three aspects:

The fix program that defines the type of change that is performed on the cataloging record

The fix routine in which the fix program runs

If required, additional parameters for the fix program

Following is a sample of the `tab_fix` table:

```
1                2                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
INS  fix_doc_tag_008
INS2 fix_doc_001
MERGE fix_doc_merge                OVERLAY-01
AUT  fix_doc_new_aut_2
```

Key to the `tab_fix` Table:

Column 1 - Routine name

Fix routines are "logical names" for defining a group of fix programs. Reserved fix routines also define when the programs are run. For example, it is possible to define a group of fix programs to run when the record is loaded to the server.

The following are ALEPH's reserved routines:

AUT

Run when the new derived authority record is created using the "derived new record" function in GUI Cataloging module.

BNA

Fix programs linked to the BNA routine are automatically activated when the Load BNA Records (file-98) batch service is used.

DEL

Fix programs linked to the DEL routine are activated automatically when a record is deleted via the GUI or during a batch process or by any other program.

HOL

Fix programs linked to the HOL routine are automatically activated when HOL records are created in the Items or Serials functions > Items list > Tab 6. (HOL Links) > "Create New" button. Note that this routine must be used in the holdings library (xxx60).

ILL-L

Fix programs linked to the ILL-L routine are automatically activated when the Locate function is activated from the ILL module (Locate button in the BIB Info tab of the ILL request).

INS

Fix programs linked to the INS routine are automatically activated when a record is sent to the server.

INS2

Fix programs linked to the INS2 routine are automatically activated when a record is sent to the server. The difference between INS and INS2 is that this routine is executed just before the record is updated in the database, and therefore it can make use of the document's system number even if it is a new document. Note that INS2 programs are run after check_doc procedures, therefore the outcome of INS2 programs is not checked before update.

INSFS

Fix programs linked to the INSFS routine are automatically activated when fast cataloging from the administrative modules (Circulation and Acquisitions). This routine is also performed when bibliographic records are created using the Special Request option in the Web OPAC and when bibliographic records are created in the Course Reading module.

LDMRG

Run automatically on a bibliographic record from MARCIVE when it is merged with a matching record in the database.

LOCAT

Fix programs linked to the LOCAT routine are automatically activated when the Locate Record function is used.

M-36

Fix programs linked to the M-36 routine are automatically performed on the records in the input file for the Check Input File Against Database (manage-36) service.

MERGE

Fix programs linked to the MERGE routine are automatically activated when the Paste Record function is used in the Cataloging module.

MNG50

Fix programs linked to the MNG50 routine are automatically activated when the Create Holdings and Item Records Using Bibliographic Data (manage-50) service is used. The programs are performed after the creation of the holdings and ADM records and can be used to modify them.

P-31

Fix programs linked to the P-31 routine are automatically activated when the Load Authority Records batch process (manage-31) is used. Currently, the `fix_doc_preferred` and `fix_doc_aut_mesh` programs should be defined in the `tab_fix` table of the authority library under the P-31 routine.

REF

Fix programs linked to the REF routine are automatically activated when the Trigger Z07 Records (manage-103) service and the Load MAB Authority Records (manage-20) are used.

UE_01

Fix programs linked to the UE-01 routine are automatically activated when the update doc daemon is activated.

The system librarian can add user-defined routine names to the lower section of the table.

Column 2 - Program name

This is the name of the program that will perform a particular fix. Each routine can have up to 20 program names assigned to it, so that a number of different fixes can be performed together. In order to assign more than one program to a routine, open a separate line for each program and repeat the routine name in column 1. For example:

```
FIX2 fix_doc_tag_008
```

```
FIX2 fix_doc_tag_100
```

```
FIX2 fix_doc_tag_250
```

In this example, whenever FIX2 is selected, three programs are run.

Column 3 - Parameters

Certain fix programs require additional information, such as table names. This column is used to define additional parameters for fix programs. Note that the documentation for each fix program indicates whether or not parameters are needed.

Following are the available fix programs:

fix_ced_uid

Creates UID fields from the 020 field or the 022 field for loading purposes (for

example, previous versions of the p_manage_18 service).

fix_doc_001

Inserts a 001 field with the system number of the record into the cataloging draft (for example, \$\$1000010091). This fix program must be attached to INS2 and not to INS, since it needs the system number of the document.

Column 3 of the tab_fix table of the library's tab directory must be used to define whether or not the program should overwrite existing 001 fields. Following are the available options:

OVERWRITE (always replaces existing 001 fields)

NO-OVERWRITE (does not replace existing 001 fields)

OVERWRITE-NON-NUMERIC (replaces only 001 field where there is at least one non-numeric character)

If no parameter is defined in the parameters column, then the default value is OVERWRITE.

Note that when the update_z103_uni linking program (UNIMARC links - Italian version) is used in the tab_z103 table of the library's tab directory, this program must be used.

fix_doc_001_prefix_sysno

This fix program automatically creates a 001 field containing a prefix defined in column 3 of the library's tab_fix table and in the system number of the record with leading zeros (for example, \$\$USM01-000003526 - in this example, USM01 has been defined in column 3 of the tab_fix table as the prefix). If a 001 field already exists, this program overrides the field and adds the new 001 field with the defined prefix.

Note that this fix program must be attached to INS2 and not INS, as it needs the system number of the document.

fix_doc_001_sysno

Automatically creates a 001 field with the library name and the system number of the record (for example, \$\$aUSM010000000000000000111142). This fix program must be attached to INS2 and not INS, for it needs the system number of the document.

fix_doc_001_sysno_inv

This routine is used to keep the original system number of records that are uploaded to the system (p_manage_18) after being exported into, for example, MARC format. When the records are downloaded into MARC format, the original system number of the records is deleted. The fix_doc_001_sysno_inv takes the system number stored in the 001 field (previously inserted by the fix_doc_sysno routine) and uses it to correct the values of the ALEPH sequential file used for uploading the records.

fix_doc_004_lkr

This routine adds the MARC 21 004 field to holdings records. The 004 field contains the system control number of the bibliographic record for which the holdings record was created.

fix_doc_005

This routine inserts a 005 field with the current date and time into the cataloging draft record. Note that it is recommended to run this program under the INS2 fix routine. This ensures that the field is created just before updating and not after reading and overriding the errors/warnings displayed in the Record Check Warnings window.

FIX_DOC_099

This routine creates:

- a 997 field with one subfield \$a for each 099 field which has two occurrences of subfield \$a, and
- a 998 field with one subfield \$a for each 098 field which has two occurrences of subfield \$a.

This can be seen in the example below:

```
099 L $$aaa $$abb -> 997 L $$aaa bb
098 L $$aaa $$abb -> 998 L $$aaa bb
```

fix_doc_035_oclc

This program copies the MARC 21 001 field (Control number) to a new MARC 21 035 field (System control number). The new 035 field is added in the following format:

```
035## $a(OCOLC)017263567
```

Note: This program does not take into account the MARC 21 003 field (Control number identifier).

fix_doc_1xx_240

The fix_doc_1xx_240 routine enables the cataloger to choose an authority heading composed from subfields \$\$a and \$\$t and to create the required 1XX/240 bibliographic combination.

If the selected 1XX field has subfield \$\$t:

- A 240 field is opened.
- The data from the subfield \$\$t of the 1XX field is copied from subfield \$\$t up to the end of the field (in other words, it includes all subsequent subfields).
- The copied data is pasted into the new 240 field with first indicator 1, changing subfield \$\$t to subfield \$\$a. All other subfields remain as they are.
- All subfields from subfield \$\$t up to the end of the field are removed from the 1XX field.

The second indicator of the created 240 field is left blank. Note that this can be automatically corrected to the correct value by using the fix_doc_non_filing_ind fix program which can be set, for example, to be run upon update of the record or as part of the fix routine to which the fix_doc_1XX_240 program is attached.

Note in addition that if the record already contains a 240 field, a new field will be created and the original field will be retained in order to enable the librarian to select the preferred field.

fix_doc_1xx_243

The `fix_doc_1xx_243` routine enables the cataloger to choose an authority heading composed from subfields `$$a` and `$$t` and to create the required 1XX/243 bibliographic combination.

If the selected 1XX field has subfield `$$t`:

- A 243 field is opened.
- The data from the subfield `$$t` of the 1XX field is copied from subfield `$$t` up to the end of the field (in other words, it includes all subsequent subfields).
- The copied data is pasted into the new 243 field with first indicator 1, changing subfield `$$t` to subfield `$$a`. All other subfields remain as they are.
- All subfields from subfield `$$t` up to the end of the field are removed from the 1XX field.

The second indicator of the created 243 field is left blank. Note that this can be automatically corrected to the correct value by using the `fix_doc_non_filing_ind` fix program which can be set, for example, to be run upon update of the record or as part of the fix routine to which the `fix_doc_1XX_243` program is attached.

Note in addition that if the record already contains a 243 field, a new field will be created and the original field will be retained in order to enable the librarian to select the preferred field.

fix_doc_880

This routine replaces the tag number of the alternate graphic representation field (880) by the associated tag registered in subfield `$6` of the field. In addition, the tag number of the associated field is removed from subfield `$6` but the occurrence number is retained. For non-880 fields, the tag number of the associated field (for example, 880) is removed from subfield `$6` and the occurrence number is retained.

This program creates parallel fields, both containing the same tag number. For example:

```
1001 L $$6880-01$$a[Name in Chinese script].
8801 L $$6100-01/(B$$aShen, Wei-pin.
```

Is changed to:

```
1001 L $$601$$a[Name in Chinese script].
1001 L $$601$$aShen, Wei-pin.
```

Note that for this fix to work, subfield `$6` must be the first subfield in both linked fields.

fix_doc_add_order_info

This routine adds to bibliographic records information from Order records (Z68) and Vendor records (Z70) attached to them. It is mostly intended for use in the UE_01 section of `tab_fix`.

For each Z68 record attached to the BIB record, fields 541 and 590 are added (or updated) as follows:

```
541###$$a - Z70-VENDOR-NAME
541###$$d - Z68-OPEN-DATE
541###$$e - Z68-ORDER-NUMBER

590###$$a - Z68-OPEN-DATE
590###$$b - Z68-APPROVER-ID
590###$$e - Z68-ORDER-NUMBER
```

Example:

```
541  L$$aJerry Books$$d20030903$$e123-1
590  L$$a20030903$$bID511260$$e123-1
```

If the bibliographic record does not contain fields 541 or 590 with Z68-ORDER-NUMBER of the processed Order record, these two fields are added. If it does, the fields are updated if necessary: 541###\$\$a is updated with Z70-VENDOR-NAME and 590###\$\$b is updated with Z68-APPROVER-ID.

Note that a bibliographic record, to which more than one Order record is attached, will have multiple occurrences of fields 541 and 590.

fix_doc_add_pinyin_check_sub9

The `fix_doc_add_pinyin_check_sub9` routine runs on fields defined in `tab_pinyin` if their content is CJK. The program takes the content of \$\$a and creates a parallel \$\$9 in pinyin, using `chi_segmentation` (Z113) and `pinyin translation` (Z114). The program can only be used in the cataloging module, with cataloger intervention.

In this program, in cases where a character has more than one pinyin option, the created subfield contains <option1, option2,...>. The cataloger can decide which to use, deleting the others.

fix_doc_add_pinyin_insert_sub9

The `fix_doc_add_pinyin_insert_sub9` routine runs on fields defined in `tab_pinyin` if their content is CJK. This routine takes the content of \$\$a and creates a parallel \$\$9 in pinyin, using `chi_segmentation` (Z113) and `pinyin translation` (Z114). In cases where a character has more than one pinyin option, the created subfield contains <option1>.

fix_doc_008_han_1

The `fix_doc_008_han_1` routine handles BIB record year information based on Chinese and Korean letters. It performs the following actions:

1. Calculates the Christian year based on the special name of the era and populates 260\$\$c/264\$\$c (publication date). If 264\$\$c is applied, it is done based on the following priority order: 2nd indicator 1,0,3,7.
2. Simultaneously updates 008/7-10 (Date 1) and 008/11-14 (Date2) according to the special name of the era in 260\$\$c. If 260\$\$c is missing, 264\$\$c is applied (based on the following priority order: 2nd indicator 1,0,3,7).

Once `fix_doc_008_han_1` is called by the system, the above actions occur and the fields `260$c/264$c`, `008/7-10` (date 1) and `007/11-14` (date 2) are updated.

fix_doc_arabic

The `fix_doc_arabic` program can be used to translate Arabic characters into the correct form (presentation forms) according to its position in the word. This fix program uses the `arabic_form` table in the `$alephe_unicode` directory for the purpose of deciding which characters to translate and the various forms. The four possible shapes of an Arabic character are:

Isolated:

The character is not linked to either the preceding or the following character.

Initial:

The character is linked to the following character but not to the preceding one.

Middle:

The character is linked to both the preceding and the following character.

Final:

The character is linked to the preceding character but not to the following one.

Following is a sample of the `arabic_form` table:

```
! 1      2      3      4      5      6
!!!!-!!!!-!!!!-!!!!-!!!!-!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
FE80                0621    ARABIC LETTER HAMZA
FE81                FE82 0622    ARABIC LETTER ALEF WITH MADDA ABOVE
FE87                FE88 0625    ARABIC LETTER ALEF WITH HAMZA BELOW
FE89 FE8B FE8C FE8A 0626    ARABIC LETTER YEH WITH HAMZA ABOVE
FEF1 FEF3 FEF4 FEF2 064A    ARABIC LETTER YEH
```

fix_doc_assign_issn

The `fix_doc_assign_issn` routine can be used by ISSN national centers. The routine creates and assigns ISSNs (022 \$a) and Centre codes (022\$2) to bibliographic records. The ISSN is composed of eight characters. The first seven characters can be the numbers 0 to 9 and the last character, which is a check character, can be the numbers 0 to 9 or an uppercase X. The first seven characters are assigned by the International ISSN Centre and the last character is calculated by the routine.

The following is an example of `./usm01/tab/tab_fix`:

```
! 1      2      3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
ISSN    fix_doc_assign_issn                $$2=9 limit=1885000
```

The `fix_doc_assign_issn` routine uses two parameters in column 3. The first parameter (`$$2`) contains the Centre code, which indicates the ISSN National Centre responsible for assigning the ISSN. The second parameter (`limit`) contains the upper limit of the ISSN counter.

For example, if `limit=1885000`, then the highest possible ISSN number is 1884-9997. The first 7 characters are the numbers of the counter and the last character is the check number.

Note

If the parameters are not defined, the `fix_doc_assign_issn` routine does not work.

Activate this mechanism by creating the create-issn counter using util G/2. The starting point of the counter is the first possible ISSN, as determined by the International ISSN Centre.

Note that if the record already contains an ISSN, the routine does not run.

fix_doc_aut_lc

This program adds subfield \$\$2 [LC] to fields 1XX, 4XX, 5XX and COR of authority records from the LC authority database.

fix_doc_aut_mesh

This program adds subfield \$\$2[MeSH] to fields 1XX, 4XX, 5XX and COR of authority records from the MeSH authority database.

fix_doc_aut_duplicate

This fix routine, which is applied in the Authority Library, deals with ambiguous headings that arise from having the same 4XX in more than one record. It does not take established headings (such as topical heading, personal name, uniform title, and so on) into account.

`fix_doc_aut_duplicate` performs the following:

- When a 4XX field is found to be duplicate to the 4XX or 1xx of another record, the 1xx field is appended to the 4XX field, using \$\$7; that is, the 4XX field is changed as follows:

```
4XX [original subfields and text] $$7 [text of 1xx (subfields are stripped)].
```

The `fix_doc_aut_duplicate` routine is based on checking the 4XX field against the ACC (Z01 headings) lists named GEN and DUP. Accordingly, these lists must be present in the authority library.s tab00 and tab01. GEN is required for authority control, and should already be present in the library.s tables. The 4##### field is sent to the DUP list, with the 7 subfield stripped. The configuration should be as follows:

```
./xxx10/tab/tab11_acc
4#####      7 *          DUP    -7

./xxx10/tab/tab00.<lng>
H DUP  ACC      11 00      0000      Duplicates

./xxx01/tab/edit_field.<lng>
```

```

1 # AUT## H -69
2           7 A ^[           ]
2           # A ^

```

If the non-preferred heading (4XX) exists in the DUP list, this means that the field already exists in other records in the Authority library. In this case, the 4XX field will be modified to include the 1XX content in the \$\$7 subfield .

If the 4XX does not exist in the DUP list, but does exist in the GEN list (meaning there is one previously existing record with the same field content), the 4XX fields of both the record being checked and of the record that is already in the database will be modified to include the 1XX content in subfield \$\$7.

If the 4XX does not exist in DUP or in GEN, it is a new heading, and there is no special treatment.

Your library can choose to use another subfield number instead of \$\$7. In order to re-configure this, change column 3 of `tab_fix` of the authorities library. `Tab11_acc` and `edit_field.<lng>` must be updated accordingly. If column 3 of `tab_fix` is empty, the default subfield is \$\$7.

The INS2 routine allows you to preview a record after the fix, before sending it to the server.

Note that it updates the existing records in the database.

The INS routine updates the record only after sending it to the server, but does not allow this preview, so that no existing records are updated before being sent to the server.

Accordingly, use the INS routine for the actual fix, and the INS2 routine for previewing fixes.

This is relevant only for the `fix_doc_aut_duplicate` routine, but not for other INS2 functions.

fix_doc_bnb

Adds to the cataloging record a 015 \$\$a field with a unique BNB (British National Bibliography) number and prefix and subfield \$\$2bnb in the same tag (for example, 015 \$\$aABC1001\$\$2bnb). The values (last BNB number and prefix) are taken from the *last-bnb-number* sequence in UTIL G/2. If the 015 field already exists, its content is not overridden.

The first place of the sequence is alphanumeric. This way it can represent higher numbers. It is numeric up to running number 99,999 and then it is changed to alphabetic (uppercase, like A, B, C and so on).

‘A’ will represent 100,000 and ‘B’ will represent 110,000 and so on.

In any one year the number assigned should run from 00001 to 99999 and then from A0000 to Z9999.

This fix program must be assigned to INS in the `tab_fix` table.

fix_doc_char_conv_z

This routine carries out character conversion using a user-defined conversion script. Column 3 can contain a conversion instance from

\$alephe_unicode/tab_character_ conversion. If Column 3 is empty, the default instance is Z. In the following example from tab_character_conversion, the Z instance uses the line_sb2line_sb procedure :

```
Z ##### # line_sb2line_sb      conversion_z
```

Here is a section from the tab_fix table:

```
CHAR    fix_doc_char_conv_z
```

fix_doc_create_035

This program moves the MARC 21 001 field (Control number) and the MARC 21 003 field (Control number identifier) to the MARC 21 035 field (System control number) deleting the original fields. The new 035 field is added in the following format:

```
035## $a(003)001
```

Additionally, this program also automatically adds to the record a new 001 field. The value for the new field is taken from the sequence "last-001-number" in UTIL G/2.

fix_doc_create_035_1

The fix_doc_create_035_1 program is similar to fix_doc_create_035. The difference is that this program does not create a 001 field (based on the last-001-number Z52 sequence) after creating the new 035 field. In short, this program can be used when you only want to create the 035 (from 001 and 003) without adding a new 001 field automatically.

fix_doc_create_066

The fix_doc_create_066 program creates the 066 field that is used to indicate the character sets present in the record. The field is created with \$c for each alternate character set. The fix_doc_create_066 program can only be used on records in the MARC-8 character set. This program is used mainly for export purposes.

fix_doc_create_fmt

This program, according to the definitions of the LDR field, adds the FMT field with the record format (for example, BK for books). The program can be used to add the FMT field to records imported through the Z39.50 server that do not have the FMT field. If the field is present, the program does nothing. This program is for use with records imported through Z39.50. Define the program in the tab_fix of the EXT library and in the \$alephe_gate/<base name>.conf file. For more information, see the *Universal Gateway* document.

fix_doc_create_hol_local_notes

This program is based on the information stored in local tags cataloged in the bibliographic record. It creates holdings records and moves the defined local tags from the bibliographic record to the appropriate holdings record. These tags can then be indexed and displayed as if they were part of the bibliographic record. The application for storing local tags in a holdings record is in a consortial environment where a single bibliographic record is shared by multiple institutions and an

institution would like to include local tags that not everyone sees. Which local tags are moved to the holding record from the bibliographic record and which are displayed in the OPAC can be configured by the local institution.

In the event that a single institution has more than one holdings record, a "super holdings" record can be created which stores local tags but not call numbers or any location information.

The following are the tables involved:

The `tab_fix` table in the `tab` directory of the bibliographic library (XXX01):

The parameters column, column 3, of the `tab_fix` table must contain the section in the `tab_fix.conf` table that specifies the local tags, the indicators, and so on. The following is a sample of the setup needed in the `tab_fix` table to use the program:

```

! 1                2                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!->
HOLD  fix_doc_create_hol_local_notes LOCAL-NOTES

```

The `tab_fix.conf` table in the `tab` directory of the bibliographic library (XXX01):

This table is used to specify the local tags, the indicators, the subfield that contains the owner of the record, the merge section used in the `tab_merge_overlay` table of the holdings library `tab` directory and the relevant section from the `tab_mapping` table of the `tab` directory of the bibliographic library. The following is a sample of the setup needed in the `tab_fix` table to use the program:

```

[LOCAL-NOTES]
local notes = 590##,690##
owners subfield = 9
owners list = AA,BB,LIN
merge section = 98
mapping section = LCN-2-HOL

```

The `tab_merge_overlay` table in the `tab` directory of the holdings library (XXX60):

This table is used to define how the holdings record is updated (merged) if it already exists. Following is a sample of the setup needed in the `tab_merge_overlay` table to use the program:

```

98 1 Y #####
98 2 Y 590##
98 2 Y 690##

```

The `tab_mapping` table in the `tab` directory of the bibliographic library (XXX01):

This table is used to define how to map the original tags from the bibliographic record into the new tags in the holdings record. The following is a sample of the setup needed in the `tab_mapping` table to use the program:

```

LCN-2-HOL      541## abcde 541   abcde      Y Y
LCN-2-HOL      541## fho39 541   fho39     Y Y
LCN-2-HOL      561## ab39  561   ab39      Y Y
LCN-2-HOL      590## ab9   590   ab9       Y Y
LCN-2-HOL      690## ab9   690   ab9       Y Y

```

Note that the original bibliographic record has the local tags taken off, but these changes do not take effect until you save the record to the server.

Note

The creation of holding record is done with no additional HOL library fix_doc programs. When the HOL record creation is initiated by the BIB library fix_doc_create_hol_local_notes, the HOL library fix programs that are defined in the HOL library tab_fix INS routines are ignored.

fix_doc_create_hol_sid

Creates SID field in the Holding record with the HOL system number in \$\$c and the related BIB in \$\$b.

fix_doc_delete_empty

This program is used to delete fields or/and subfields that do not have any content. Note that in the Cataloging module, when a record is saved on the server, empty fields/subfields are deleted automatically. This fix program can be used, for example, when cataloging records are loaded by other services (for example, cataloging loaders) that do not perform this deletion automatically.

fix_doc_delete_chi_spaces

This program is used to delete spaces between two Chinese letters.

fix_doc_field_200

This program can be used by UNIMARC libraries to create additional fields based on the 200 field (title and statement of responsibility).

If there is more than one \$a subfield in field 200, the system creates 423 fields, one for each occurrence of subfield \$a, except for the first one. In the 423 field, the first indicator will be blank and its second indicator will be zero. The \$a subfield of the 423 field will contain the embedded content of the 200 field's \$a subfield. The \$c subfield of the 423 field will contain the embedded content of the 200 field's \$c subfield that follows \$a in the 200 field. The 423 field will also include a \$1 subfield containing a 200## tag.

Here is an example:

```
423_0 $a (The second $a in field 200)
      $c (Optional, contains the second $c of field 200)
      $1 (200##)
```

The program also creates a new 510 field \$a for each occurrence of \$d in the 200 field. The first indicator of field 510 is 1 and the second one is blank.

Here is an example:

```
5101_ $a ($d in field 200)
```

This fix program does not delete previously created 423 and 510 fields.

fix_doc_field_410

This program can be used by UNIMARC libraries to create an additional field based on field 410#1 (Series).

If a UNIMARC 410#1 field contains \$a and \$1, the program will create a 2252# field. \$a of the 2252# field will contain the embedded content of the 410 \$a field.

Here is an example:

```
2252# $a<$a in field 410#1>
```

The fix program does not delete previously created 2252# fields.

fix_doc_field_lower / fix_doc_field_upper

The **fix_doc_field_upper** and **fix_doc_field_lower** programs change the case of all occurrences of a field given as a parameter to the routine. The **fix_doc_field_upper** program changes the case to uppercase and the **fix_doc_field_lower** program changes the case to lowercase.

The parameter has the following format: *<field><tag><subfield>*

Here is an example from the tab_fix table of a UNIMARC library:

```
! 1                2                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
LOWER fix_doc_field_lower                701#0a
```

As shown in this example, the # sign must be used if no character is defined for the field or for the indicator.

In order to change every subfield in a field, you do not have to add a subfield parameter. If a subfield parameter is present, the case of all its occurrences will change accordingly.

fix_doc_do_file_08

The **fix_doc_do_file_08** program is a generic fix program that modifies cataloging records based on a supplied processing script. Many standard fix programs are provided by ALEPH, but there are times when a library would like to perform a customized fix on a record. This can be done by the **fix_doc_do_file_08** program.

The processing script must be located in the tab/import directory of the library. The table name is user-defined. You can create multiple tables to define different fix procedures. The script is in the format of a normal ALEPH table with 9 columns.

Note that the maximum number of lines that the file can contain is 500 lines.

Note:
A line should not exceed 155 characters, even a comment line (starting with "!").

The `generic_fix` table in the `USM01 $data_tab/import` directory is an example of a processing script. Specifications for the conversion script can be found in the [41 Setting Up a Script for the Correction of Records in Aleph Sequential Format](#) chapter on page [116](#) in this document. The following is a sample of the table:

```
! 2 3 4 5 6 7 8 9
!-!!!!-!!-!!-!!-!!-!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!>
1 001 COPY-SYSTEM-NUMBER 035 ,L,a
1 035## REPLACE-STRING ocm,
(OCoLC)
1 LDR ADD-FIELD OWN
,L,$$LIN
2 852## CHANGE-FIELD 949
3 949## DELETE-SUBFIELD c
3 949## REPLACE-STRING $$i,^
3 949## CHANGE-SUBFIELD h c
3 949## CHANGE-SUBFIELD b h
```

fix_doc_fixed_fields

This program replaces the hyphens ("-") in fixed-length fields (such as MARC 21 LDR, 006, 007 and 008 fields) by a caret ("^"). This program can be used to correct MARC 21 records in which blanks in fixed-length fields are marked by hyphens (hyphen is a valid value in MARC 21 and should not be used to mark spaces).

fix_doc_hld_stmt

This program performs the same procedures as the `expand_doc_hld_stmt` program. The difference between the two programs is that when running the `expand_doc_hld_stmt` program the 863/4/5 enumeration and chronology data are created virtually, whereas when running `fix_doc_hld_stmt`, the 863/4/5 enumeration and chronology data are added to the HOL record.

For the `fix_doc_hld_stmt` program, the `tab_fix` table of the HOL library (XXX60) should include the following line:

```
! 1 2 3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
HSTMT fix_doc_hld_stmt
```

To enable the generation of summary holdings statements for item records that are not linked to a subscription record or do not have a value in the Copy ID field in the subscription record, the parameter:

```
GET_Z30_BY=HOL
```

should be defined in col.3 of `tab_fix` as follows:

```
! 1 2 3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
HSTMT fix_doc_hld_stmt GET_Z30_BY=HOL
```

For item records which are linked to a subscription record and which hold a value in the Copy ID field in the subscription record and the item record, the parameter:

```
GET_Z30_BY=COPY_ID
```

can be defined in col.3 of tab_fix as follows:

```
!      1                      2                      3
!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
HSTMT      fix_doc                      GET_Z30_BY=COPY_ID
```

Note: If no parameter is defined in col.3 of tab_fix the system will use the default parameter:

```
GET_Z30_BY=COPY ID
```

For full details of this program refer to expand_doc_hld_stmt in the Expand Routines section of the Indexing chapter.

fix_doc_hol_852_1

This program inserts a space between the main class and the first cutter of the call number, and was required in former versions for filing consistency. The fix procedure is no longer required, since the "lc_call_no" filing procedure creates the same filing key, whether or not the space is present. The space is inserted if the first cutter is held in subfield \$h of the MARC 21 852 field and if there is not already a space.

For example, this program changes:

```
LOC0 L $$bMAINX$$cLIB$$hE183.8.B7$$iW45
1993$$oBOOK$$4MAIN$$5LIB$$3Book
to:
```

```
LOC0 L $$bMAINX$$cLIB$$hE183.8 .B7$$iW45
1993$$oBOOK$$4MAIN$$5LIB$$3Book
```

fix_doc_ldr_05_d

fix_doc_ldr_05_d sets the value of position 05 of the LDR field of the record to *d*. It can be listed under the DEL section.

fix_doc_ldr_sta_delete

The fix_doc_ldr_sta_delete program changes the record status in position 05 of the LDR field to 'd' if the STA \$\$aDELETED field is present in the record.

fix_doc_lkr_up

This program adds subfield \$n (up link note) and subfield \$m (down link note) to the LKR field of types UP (up link), ANA (analytic) and PAR (parallel). Subfield \$m is built based on the 245 field of the record with the LKR field. Subfield \$n is built based on the 245 field of the record to which the LKR points. If the records are for a serial (FMT = SE), then subfield \$n and subfield \$m are built from the corresponding 222. If the 222 field is not present, then the subfields are built from field 245.

It is possible to exclude punctuation suffixes by using the DELSUF parameter in column 3 of tab_fix. This parameter specifies the punctuation marks to be excluded.

The ./xxx01/tab/tab_fix setup example below specifies which punctuation suffixes (slash, space + period, and space+ colon) to exclude from subfields \$m or \$n of the LKR field when text is copied by fix_doc_lkr_up from 245\$a or 245\$b:

```
! 1                2                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
INS  fix_doc_lkr_up                DELSUF=|/|^.|^:|
```

Notes:

- Within the parameter (key=value) no space is allowed, because spaces are often used as delimiters for different parameters.
- If a space should be part of the suffix, use a caret (^).
- The first sign of the value is the delimiter of different suffixes. Use a character that is not used as any suffix of those listed. In the example above, the delimiter is the pipe sign (|).

fix_doc_7xx_lkr

The fix_doc_7xx_lkr fix routine supports the 7XX field update based on information from the LKR field.

For example: ./usm01/tab/tab_fix

```
! 1                2                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
7XX  fix_doc_7xx_lkr
```

For example: ./usm01/pc_tab/catalog/fix_doc.eng

```
! 1  2 3                4
!!!!-!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
7XX  N L Create 7XX based on LKR field
```

fix_doc_7xx_lkr creates the 7XX field based on the LKR field.

The following is a description of the fix activation flow, including the cataloger's manual actions and the fix's automatic actions:

The cataloger performs the following:

1. The cataloger catalogs LKR \$\$b (the system number of the linked record) and \$\$r (the 7XX tag that contains the information of the linked record). If necessary, the

cataloger also catalogs subfield \$\$z (to indicate the type of relationship) and \$\$g (to indicate the enumeration and chronology of the issue in which the absorption or split occurred).

2. The cataloger activates the new fix either manually by selecting the relevant fix routine or automatically when saving the record.

The fix performs the following:

1. The fix checks the FMT of the record.
 - If FMT is not SE or 902\$\$b = 631, 632, 633, or 634– no activity is performed.
 - Otherwise, the fix continues with the next step.
2. The fix first adds subfield \$\$a with the value PAR and subfield \$\$l with the value <BIB library> to the LKR field. (<BIB library> represents the BIB library code – for example – USM01.)
3. If LKR\$\$aPAR has been added previously, the fix enriches the LKR fields by creating the \$\$x, \$\$n, and \$\$m subfields.
 - Subfield \$\$m is created from the concatenation of subfields \$\$a, \$\$n, \$\$p, \$\$h, and \$\$c of the 245 field of the record being edited. If a subfield \$\$m already exists in the LKR field, it is not replaced.
 - Subfield \$\$n is created from the concatenation of subfields \$\$a, \$\$n, \$\$p, \$\$h, and \$\$c of the 245 field of the linked record. If a subfield \$\$n already exists in the LKR field, it is not replaced.
 - Subfield \$\$x is created from the value of the first 022\$\$a of the linked record. If a \$\$x subfield already exists in the LKR field, it is not replaced. If subfield \$\$x includes only a hyphen (-) (indicating that it should be treated like an empty subfield), the value is not copied from the linked record.
4. The fix deletes the following 7XX fields that have a subfield \$\$w: 765, 767, 775, 776, 777, 780, 785, 786, and 787. A 7XX field without a \$\$w subfield is left unchanged.
5. If the LKR contains subfield r, the fix automatically creates one 7XX field for each LKR of type PAR.
 - The tag of the 7XX field is the content of LKR \$\$r. (The field name is created from the first three digits of the value of LKR \$\$r).
 - The first digit of the indicator is created from the fourth digit of the value of LKR \$\$r.
 - The second digit of the indicator is created from the fifth digit of the value of LKR \$\$r. If the fifth digit of the value of LKR \$\$r is blank, the second digit of the indicator is also blank.

- \$\$w contains the 001 number of the linked record (not the system number)
- \$\$t is created by concatenating the \$\$a, \$\$n, \$\$p, \$\$h, and \$\$c of the 245 field of the linked record.
- \$\$x is copied from LKR \$\$x
- \$\$i is copied from LKR \$\$z
- \$\$g is copied from LKR \$\$g

fix_doc_lng_from_bib

The `fix_doc_lng_from_bib` program can be used to update the language code in the 008 field of the holdings record and/or ADM record according to the language code in positions 35-37 of the 008 field of the associated bibliographic record. In the ADM record, positions 35-37 of the 008 field are updated with the language code. In the holdings record, positions 22-24 are updated with the language code.

Note that this fix is not required for the online update of the language code in the 008 field of the holdings records. Holdings records created in the system are updated according to the `HOL-008-LNG` variable of the `tab100` table of the holdings library (for example, `USM60`). This variable is used to determine the default language code for the MARC 21 008 field in holdings records:

If the variable is set to 0, then the language code of the 008 field of the holdings record is set to the defaults specified in the `tab_tag_text` table.

If the variable is set to 1, the language code of the 008 field of the holdings record is taken from the bibliographic record based on standard system rules (that is 008, 041, and so on).

fix_doc_marc21_spaces

This program converts contiguous runs of two or more blanks (spaces) into the blank character specified by the `DOC-BLANK-CHAR` variable in the library's `tab100` table. The following MARC 21 fields are affected by the fix: 010##, 260##, 310##, 321##, 362##, 515##, 525##, 533##, 76###, 77###, and 78###.

fix_doc_merge

This program merges or overlays cataloging records according to the merging program defined in the `tab_merge` table located in the library's `tab` directory. Column 3 of the `tab_fix` table is used to define the merging routine that matches the relevant section in the `tab_merge` table. Refer to *Merging Records* on page 84 for more information.

fix_doc_ndl_nb_br

This fix automatically creates the Japanese National number and the BR number in bibliographic records. The fix is based on the information stored in local field 900\$a cataloged in the bibliographic record. There are three types of numbers that are differentiated by their prefix. Each type has its own counter. Util G/2 defines the `ndl-nb-2`, `ndl-nb-0`, and `ndl-br-1` counters. Column 3 of `tab_fix` is used to define the upper limit of the Japanese National number and the BR number:

The following is an example of `./ndl01/tab/tab_fix`

```
! 1                2                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
NBBR   fix_doc_ndl_nb_br                nb2=9999999 nb0=9999999 br1=9999999
```

- nb2 – indicates the upper limit of Japanese National Bibliography number. with a prefix of 2
- nb0 – indicates the upper limit of Japanese National Bibliography number. with a prefix of 0
- br1 – indicates the upper limit of BR number with a prefix of 1.

Note that this fix must be manually activated.

fix_doc_new

This fix routine provides a generic platform for deriving a new record, based on a previously cataloged record. A companion configuration table, `tab_fix_new`, is used to define which tags should be generated for the new derived record. It is possible to define:

- tags based on tags of the source record; a single new tag can be composed of several tags from the source record
- tags with content, including fixed length fields
- empty (template) tags - and similar tags
- a program in column 7 of `tab_fix_new`
- manipulate tags

`tab_fix_new` can have several sections; a specific line in `tab_fix` is linked to the relevant `tab_section` by defining "TYPE=" in col. 3. For example, if the relevant section of `tab_fix_new` is called "son", then col. 3 of `tab_fix` will have TYPE=son.

Refer to the table's header and Configuration Guide documentation for more information on the `tab_fix_new` table.

fix_doc_new_ana

This program creates an analytic record from the current record. The new record is created as follows:

LDR: Position 07 is set to b. Other default values are defined according to the specifications of the `tab_tag_text` table of the library's tab directory.

008: Default values are defined according to the specifications of the `tab_tag_text` table of the library's tab directory.

050: If the field exists in the parent record, it is taken to the new record.

080: If the field exists in the parent record it is taken to the new record.

245: The field is opened but remains empty.

260: The field is taken from the parent record.

300: Subfield c is taken from the parent record.

LKR: Points to the parent record.

fix_doc_new_aut_2

This program is used by MARC 21 libraries to create an authority record from the current bibliographic record. The new record is created as follows:

If the 100 field is present in the bibliographic record, then the authority record will be derived with a 100 field.

If the 110 field is present in the bibliographic record, then the authority record will be derived with a 110 field.

If the 111 field is present in the bibliographic record, then the authority record will be derived with a 111 field.

If the 130 field is present in the bibliographic record, then the authority record will be derived with a 130 field.

The 670 field of the authority record is created from the 245 (subfield \$a) and 260 (subfield \$c) fields of the bibliographic record. This program also adds text to the new 670 field. The text can be configured via a message file in \$aleph_root/error_lng called: fix_doc_new_aut_2.

The record created by the program is by default set to the authority library defined under the AUT section of the library_relations table. If this section is not present, you can define the authority library by using the parameters column of the tab_fix table. If no authority library is defined in the library_relations table and in the tab_fix table, the system uses the default of XXX10 as the authority library (this is done by setting the last two digits in the active library as 10).

fix_doc_new_aut_3

This program is used by UNIMARC libraries to create an authority record from the current bibliographic record. The new record is now created as follows:

If the 70# field is present in the bibliographic record, then the authority record is derived with a 200 field.

If the 71# field is present in the bibliographic record, then the authority record is derived with a 210 field.

If the 72# field is present in the bibliographic record, then the authority record is derived with a 220 field.

If the 73# field is present in the bibliographic record, then the authority record is derived with a 200 field.

The 810 field of the authority record is created from the 200 field - subfield \$a and from 210 - subfield \$c and \$d. This program also adds text to the new 810 field. The text can be configured via a message file in \$aleph_root/error_lng called fix_doc_new_aut_3.

The record created by the program is by default set to the authority library defined under the AUT section of the library_relations table. If this section is not present,

you can define the authority library by using the parameters column of the `tab_fix` table. If no authority library is defined in the `library_relations` table and in the `tab_fix` table, the system uses the default of XXX10 as the authority library (this is done by setting the last two digits in the active library as 10).

fix_doc_new_aut_4

This program is used by UNIMARC libraries to create an authority record from the current bibliographic record. This program is cursor-sensitive and according to the position of the cursor, the new authority record is created either from the relevant 6XX, 5XX, or 7XX fields (see complete list below). The new authority record is created as follows:

The original field (for example, 700) is taken as is and placed in the corresponding 2XX field in the authority record.

A new 810 field is built based on the 200 / 230 / 250 fields (subfield \$a) and from the 210 field (subfields \$c and \$d) of the bibliographic record. The new program also adds text to the new 810 field. The text can be configured via a message file in `$aleph_root/error_lng` called `fix_doc_new_aut_4`.

For example, if the bibliographic record contains the following fields:

```
200 1#$Steam locomotives of Germany and Austria
210 ##$a[Cambridge, Mass.]$cHarvard Univ. P.$dl981
```

Then the new 810 field is added as follows in the derived authority record:

```
810 ##$aSteam locomotives of Germany and Austria, Harvard Univ., 1981
```

In addition, the fields 152 (with subfields \$a and \$b) and 801 (with subfields \$a, \$b and \$c) are added to the record without contents.

The record created by the program is by default set to the authority library defined under the AUT section of the `library_relations` table. If this section is not present, you can define the authority library by using the parameters column of the `tab_fix` table. If no authority library is defined in the `library_relations` table and in the `tab_fix` table, the system uses the default of XXX10 as the authority library (this is done by setting the last two digits in the active library as 10).

Note that if the cursor is not placed on one of the relevant tags (6XX, 5XX, or 7XX fields), then the authority record derived from the bibliographic record is created with the default LDR and 008 fields and without the 2XX, 152, 801 and 810 fields.

The following is the list of relevant fields for the creation of the authority record:

500, 600, 601, 606, 700, 701, 702, 710, 711, 712, 720, 721, 722 and 730.

fix_doc_new_aut_5

This program is used by MARC 21 libraries to create an authority record from the current bibliographic record. This program is cursor-sensitive and according to the position of the cursor, the new authority record is created either from the relevant 1XX, 4XX, 6XX or 7XX fields (see complete list below). The new authority record is created as follows:

The original field (for example, 600) is taken as is and placed in the corresponding 1XX field in the authority record.

A new 670 field is built based on the 245 (subfield \$a) and 260 (subfield \$c) fields of the bibliographic record. The new program also adds text to the new 670 field. The text can be configured via a message file in \$aleph_root/error_lng called fix_doc_new_aut_5.

The record created by the program is by default set to the authority library defined under the AUT section of the library_relations table. If this section is not present, you can define the authority library by using the parameters column of the tab_fix table. If no authority library is defined in the library_relations table and in the tab_fix table, the system uses the default of XXX10 as the authority library (this is done by setting the last two digits in the active library as 10).

Note that if the cursor is not placed on one of the relevant tags (1XX, 4XX, 6XX or 7XX fields), then the authority record derived from the bibliographic record is created with the default LDR and 008 fields and without the 1XX and 670 fields.

List of relevant field for the creation of the authority record:

100, 110, 111, 130, 440, 490, 600, 610, 611, 630, 650, 651, 700, 710, 711, 730 and 740.

fix_doc_new_aut_6

This program is used by MARC 21 libraries to create an authority record from the current bibliographic record. This program is cursor-sensitive, and is equivalent to the fix_doc_new_aut_5 program, except for the following changes:

- No additional \$aleph_root/error_lng/fix_doc_new_aut_5 text is inserted in the 670 field.
- The title and date in the \$\$a subfield of the derived 670 field are separated by a comma and a space.
- The \$\$a subfield of the derived 670 ends with a colon. If there is a period at the end of the derived 670 field, it is removed.
- The record may be derived also from the 8XX fields. Deriving from the 800, 810, 811, and 830 fields is the same as from the 700, 710, 711, and 730 fields respectively.

fix_doc_new_aut_7

This fix_doc program is used by MARC 21 libraries to create an authority record from the current bibliographic record (using the "derive new record" function). The new authority record is created based on 1xx, 6xx, and 7xx (same as fix_doc_new_aut_5 routine). fix_doc_new_aut_7 works also for subject fields cataloged in 69x fields (Hebrew subjects). With this new program, the authority record created will contain the text from the 69x field in the 159 field.

fix_doc_non_filing_ind

Automatically inserts the relevant non-filing indicators according to the stopwords

defined in the tab02 table of the library's tab directory. Non-filing indicators for each tag are defined in the library's tab01.lng table, column 6.

fix_doc_non_filing_ind2

This program is similar to fix_doc_non_filing_ind but can also process stopwords in tab02 that contain an apostrophe or a space (for example: "n" or "na h-").

When defining a non-filing indicator in tab02 that contain an apostrophe or a space, use fix_doc_non_filing_ind2 instead of fix_doc_non_filing_ind.

fix_doc_notes

Replace text with an alternative text. Enables the automatic translation of bibliographic note fields (for example, the: translation from English to French). This fix works in conjunction with the tab_fix_notes Aleph table (list of translations per bibliographic tag and subfield). For more information, refer to [Automatic Translations – Functionality and Examples](#) on page 181.

fix_doc_oclc

The fix_doc_oclc program is used for the OCLC record loader. The program moves the OCLC 001 and 003 fields to the ALEPH (MARC 21) 035 field, in the following format: (003)001.

fix_doc_oclc_2

This program is similar to fix_doc_oclc, except that it deletes any existing 035 field in the incoming record before writing the 001/OCLC number to a new 035 field. The fix also adds the UPD field (Y or N) to authority records.

fix_doc_oclc_retain_001

This program is similar to fix_doc_oclc and fix_doc_oclc_2, except that this procedure adds the new 035 field without deleting the OCLC 001 field or the 003 field.

fix_doc_overlay

The fix_doc_overlay program allows the user to merge records when they are uploaded to the system. It is used to upload those records that find a unique match in the database when the Check Input File Against Database (manage-36) service is run. The records in the output file produced by this service are given new system numbers that match the system numbers of the corresponding records found in the database. The fix_doc_overlay program can be used to merge the records in the database with the new incoming records using a merge routine from the tab_merge_overlay table. Column 3 of the tab_fix table is used to define the merging routine that matches the relevant section in the tab_merge table. When running the Load Catalog Records (p-manage-18) function, select the fix routine defined for this program.

fix_doc_own_1

The fix_doc_own_1 program inserts the value of the Cat. OWN ID field of the cataloger to the OWN field of new created record.

fix_doc_preferred

Automatically creates a COR field when the preferred term of the authority record is changed. The COR field contains the original term. This is necessary so that the link with bibliographic records (that have the original preferred term) is retained.

fix_doc_punctuation_usm

Adds punctuation to MARC 21 245, 260, 264, and 300 fields. The program only deals with subfields a, b and c of the fields.

If the last subfield ends in ".", "!" or "?", a full stop "." is not added to it. The program can specify punctuation marks in column 3. Here is an example:

```
! 1                2                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
INS  fix_doc_punctuation_usm          .!?-
```

If the last subfield of one of the these fields - 245, 260, 264, and 300 fields. - ends with one of the punctuation marks specified in column 3 example, a full stop will not be added to it.

If column 3 is empty, in order to add punctuation marks to the three default ones, the default marks (.!?) must be specified in column 3 together with other marks. For instance, a hyphen can be added as shown in the above example.

fix_doc_qualified_ucs

This fix_doc routine accepts a list of fields as parameters. The routine checks the existence of the fields and updates or creates the QUA field, as follows:

- If the fields exist, then the single subfield \$\$a of the field is set to Y.
- If the fields exist and the cataloger is identified as UCS staff (the Cataloger's OWN ID is 'OLCC'), the single subfield \$\$a of the field is set to A
- If at least one of the fields does not exist, the single subfield \$\$a of the field is set to N

For example, consider the following tab_fix setup:

```
INS2  fix_doc_qualified_ucs  245##,1001#
```

In the above example, the QUA field is updated with "Y" only if both fields 245## and 1001# exist. If the updating cataloger is UCS staff, the status is automatically set to "A".

If either field is absent, the QUA field is created or updated with N.

The maximum number of allowed fields in this table is 16.

fix_doc_redo_880

This routine reverses the effects of the fix_doc_880 program. This program restores the tag number of the alternate graphic representation field (880). For example:

```
1001 L $$601$$a[Name in Chinese script].
1001 L $$601$$aShen, Wei-pin.
```

Is changed to:

```
1001 L $$6880-01$$a[Name in Chinese script].
8801 L $$6100-01/(B$$aShen, Wei-pin.
```

Note that the order of the paired fields is important, because the tag of the first of a pair is left as is, and the second of a pair is transferred to 880.

In addition, note that the input for this program must be in MARC8 (not in UTF) encoding. The reason for this is that this fix routine sets the escape sequence and orientation for the language code, and in order to do so, the record must be in MARC8 encoding. The `fix_doc_redo_880` program will work correctly on UTF records, but will not set the escape sequence and orientation for the language code.

fix_doc_ref_1

This program is used to update a non-preferred term in the bibliographic record to a preferred term. The correction occurs only if the UPD field in the authority record is set to "Y". The `tab_fix` table of the bibliographic library must include the `fix_doc_ref_1` program under all relevant sections:

```
INS    fix_doc_ref_1
UE_01  fix_doc_ref_1
REF    fix_doc_ref_1
```

The `fix_doc_ref_1` program must be included under INS for the update of records from the Cataloging module; under UE_01 for the indexing daemon (UE_01 process); and, if necessary, under REF for the Trigger Z07 Records (manage-103) service.

Column 3 of the `tab_fix` table may be used to set a Y\N\A parameter.

For example: `./usm01/tab/tab_fix`

```
! 1                2                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
UE_01 fix_doc_ref_1                Y,A
```

‘Y’ and ‘N’ parameters set the library’s policy in case the bibliographic heading record and the authority heading record are not of the same language script, i.e. one is Latin and one is not. If the parameter is set to Y then the link to the authority record will be created, but the update of the bibliographic record will not take place. The parameter’s default is N.

In addition, if the bibliographic heading contained the non-preferred form of the heading and if the conditions for bibliographic updated exist (UPD field is 'Y' and `fix_doc_ref_1` has been defined in the `tab_fix` table), then the bibliographic record is updated as in the following example:

Authority record:

```
150 $$aFighting dogs
450 $$aPit dogs
```

Bibliographic record:

```
65010 $$aPit dogs $$zItaly
```

The bibliographic record is updated as follows:

```
65010 $$aFighting dogs $$zItaly
```

If column 3 of tab_fix is set with 'A'; the flip of the bibliographic heading takes place only if there is a match to the authority preferred term. The update of the bibliographic heading (1XX field) based on authority non-preferred term (4XX field for example) does not occur at all. The link to the authority record is created in any case.

Manual update of bibliographic heading base on the authority 4XX heading is enabled, that is, if a user manually searches for a heading to a bibliographic record (via GUI-Cataloging, CTR+ F3), he/she is able to copy the 4XX non-preferred term to the bibliographic 1XX heading.

If tab_fix is set with both parameters 'Y' and 'A', there is also a check for a match in the language script (the update of the BIB 1XX occurs only if it is in the same language script as the AUT 1XX).

Column 3 of the tab_fix table may be used to set a Y\N parameter that sets the library's policy in case the bibliographic heading record and the authority heading record are not of the same script, that is, one is Latin and one is not. If the parameter is set to Y then the link to the authority record is created, but the update of the bibliographic record dose not take place. The parameter's default is N.

To deactivate the automatic updating of subfield \$\$6 in the BIB record from the references of the Authority record, set column 3 of the tab_fix table with the parameter J.

fix_doc_rlin_1

This program moves the MARC 21 001 field (Control number) and the MARC 21 003 field (Control number identifier) to the MARC 21 035 field (System control number), deleting the original fields. The new 035 field is added in the following format:

```
035## $a(003)001
```

fix_doc_shelf_mark

Intended for use with HOL records. It appends a counter to the \$\$j subfield of the 852 field. The counter is based on a prefix that is cataloged in the \$\$j subfield and the 'bl' prefix. For example, consider that the following subfield is cataloged in the 852 field:

```
$$j mss
```

If the HOL library has a counter of 'bl-mss' with the value 12, the fix_doc_shelf_mark routine will fix the \$\$j field to \$\$j mss.13

fix_doc_sort

Sorts the fields of the current record according to the order defined in the ALEPH table of codes (tab01.lng). However, within the MARC 21 5xx, 6xx and 7xx groups of fields, the order of the fields remains as they were entered by the cataloger. Note that for the sorting of the 5xx, 6xx and 7xx blocks, the 500, 600, 700 and 800 codes must be explicitly listed in the table even if they are not used. Fields that do not have any content are deleted. In an authority library, all fields are sorted according to the order defined in the ALEPH table of codes (tab01.lng).

fix_doc_sort_505

This program is similar to fix_doc_sort, except that it sorts the 505 fields as a group after all of the other 5XX fields.

By using this program, the 505 fields are inserted as one block, not interspersed with the existing 5XX fields block. The order of the 505 fields remains as they were entered by the cataloger.

fix_doc_sort_lkr

This program sorts the LKR fields of the record in the following order: DN, PAR, UP. Note that for this program to run, it is also necessary to define the fix_doc_sort program that is used to sort the fields of the record.

For the fix_doc_sort_lkr program, the tab_fix table should include the following lines:

```
! 1                               2                               3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
INS  fix_doc_sort
INS  fix_doc_sort_lkr
```

* Note that INS is used here as an example. The fix can be attached to any routine name (reserved or user-defined).

fix_doc_sort_sub6

This program should be used after performing the fix_doc_880 program that creates parallel fields that are different script representations of each other. The fix_doc_sort_sub6 program is used to sort the linked-parallel fields. The fields are sorted by the occurrence number stored in subfield \$6.

fix_doc_space_char

This program changes a character that has been used as a placeholder for blanks to a blank, in the fixed fields LDR, 001-008 in MARC21 and LDR, 001, 005, 100 in UNIMARC. This can be used, for example, in the Download Machine Readable Records (print_03) service when exporting records in MARC format.

You define the character that acts as a placeholder for blanks in tab100 using the DOC-BLANK-CHAR variable. Only the character defined will be replaced by a blank. For example, in USM01, the caret character is defined as the "BLANK-CHAR".

You can include or exclude specific fields, by using column 3 of tab_fix option (replacing the tab100 variable DOC-BLANK-CHAR with a space).

The parameters in column 3 of tab_fix are comma-separated fields (five characters each, which can include hashes, for example, 10####) to include or exclude (to exclude, prefix the list of fields with a SINGLE dash (-), that is -010##,100##,245##). Note that the total length of the fields list must not exceed the length of column 3 of tab_fix, specified in the table's header.

The following is an example of fix_doc_space_char in tab_fix with inclusion (in this example, ONLY the specified fields will be processed):

```
! 1                               2                               3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
EXPRT fix_doc_space_char          010##,008##,001##
```

The following is an example of fix_doc_space_char in tab_fix with exclusion (in this example, ALL record fields will be processed EXCEPT from the specified fields):

```
! 1                               2                               3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
EXPRT fix_doc_space_char          -010##,008##,001##
```

If column 3 of tab_fix for fix_doc_space_char is left BLANK, the routine will behave as follows:

If the tab100 variable MARC-TYPE = "1" (USMARC), the following fields are processed: LDR and 001 - 008.

If MARC-TYPE = "2" (UNIMARC), the following fields will be processed: LDR, 001, 005, 100##

Otherwise (when MARC-TYPE is other than "1" or "2"), only LDR will be processed.

fix_doc_sub

This program adds subfield \$\$2[MeSH] to fields 6XX of bibliographic records. The subfield is only added when the second indicator of the field is 2 (Medical Subject Headings).

fix_doc_suppress

This fix routine checks the bibliographic record according to the check_doc section that is defined in column 3 of the tab_fix line. If the check succeeds, a STA field is added to the record with the value SUPPRESSED. If the check fails, the STA field with the SUPPRESSED value is removed. This routine may be used in the UE_01

section of tab_fix if automatic suppression of records is wanted when a BIB record has no attached ADM information. The following are examples of ADM Information: Items, Orders Subscriptions, Links to other BIB, Links to HOL records, and other deletion checks.

fix_doc_tab04_(01-99)

Translates the field codes of the record into another set of field codes. The translation values are defined in the library's tab04 table. The suffix defines which set of codes is chosen from the table (for example, fix_doc_tab04_01 refers to set 01 in tab04).

fix_doc_tag_008

Automatically adds the date of publication to positions 07-10 (or corrects the existing values) of MARC 21 008 field according to the date entered in MARC 21 260 field, subfield \$c. If 260\$\$c is missing, 264\$\$c is applied (based on the following priority order: 2nd indicator 1,0.3,7).

fix_doc_tag_100_open_date

Automatically adds the current date (creation date) to positions 00-07 of UNIMARC 100 field. The date is entered in the pattern YYYYMMDD.

fix_doc_tag_008_open_date

Automatically adds the current date (creation date) to positions 00-05 of MARC 21 008 field.

fix_doc_tag_008_update_date

This fix routine enables the adding of the update date to positions 26-31 of an HOL record's 008 field.

fix_doc_tag_041

This fix program updates positions 35-37 in field 008 with the language that is set in subfield \$\$a of field 041.

fix_doc_transliteration

If activated in a BIB library, this program creates an additional occurrence of the field for which it was called, entering transliterated contents of the source field into this new parallel field. The program works with routines TRNL1, TRNL2, TRNL3, TRNL4, TRNL5 and TRNL6. Each routine works with program arguments pointing to a transliteration table or program. This routine is currently available for the CJK contents only.

TRNL1	fix_doc_transliteration	HANJA_TO_HANGUL
TRNL2	fix_doc_transliteration	HANJA_TO_PINYIN
TRNL3	fix_doc_transliteration	PINYIN_TO_HANGUL_MOE
TRNL4	fix_doc_transliteration	PINYIN_TO_HANGUL_CK
TRNL5	fix_doc_transliteration	KANA_TO_ROMANIZED_KANA
TRNL6	fix_doc_transliteration	KANA_TO_HANGUL

- HANJA_TO_HANGUL - transliterates Hanja characters into Hangul
- HANJA_TO_PINYIN - transliterates Hanja characters into Pinyin

If multiple possible transliterations are detected, the fix routine puts them in subsequent double square brackets, with the possible transliterations of each term separated by commas within the square brackets.

For example

```
700$$a[[zhong,guo]] [[bao,xian]] [[jian,du,guan]] li [[wei,yuan,hui]]
```

- FIX_HANJA_TO_PINYIN – fixes the transliteration that is created by HANJA_TO_PINYIN

Removes the double square brackets that are created by the HANJA_TO_PINYIN option and removes all the transliteration options for each term except the first one.

- PINYIN_TO_HANGUL_MOE and PINYIN_TO_HANGUL_CK - both transliterate Pinyin input into Hangul, but the mappings are slightly different.
- KANA_TO_ROMANIZED_KANA - transliterates Japanese Kana entries into Latin-alphabet Kana
- KANA_TO_HANGUL - transliterates Japanese Kana into Hangul

If activated in the AUT library, this fix routine transliterates the contents of \$\$q of the 1XX entry and enters the resulting string into a new occurrence of field 400, subfield \$\$a. The settings in tab_fix in the AUT library are the same as in the BIB library.

fix_doc_trans_doc

This fix routine adds translated content of specific fields. It works with a configuration table that specifies:

- Which fields in the document are translated
- Which target field stores the translated content
- Which translation method is used

The configuration table is sent to the fix routine as a parameter.

For example, [BIB library]/tab/tab_fix:

```
! 1                2                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
TRNL1 fix_doc_trans_doc                tab_doc_trans
```

The configuration table must exist in the library where the fix is performed (the Bibliographic library or the AUT library) and have the following structure:

Column 1: source field and indicators

Size: 5

can be used for the fourth and fifth positions to indicate truncation of numeric additions to the field code (for example, 245## for 2451, 2452, 24501)

Column 2: source subfield(s)

Size: 10

- Transliteration that is created by HANJA_TO_PINYIN is translated using PINYIN_TO_HANGUL_CK

The translated content will be stored in field 940.

fix_doc_uk_222

This program adds a new 222 UKMARC field to SE (serial) format records. The 222 field is created from fields 245 and 240, subfields \$a, \$j and \$s.

fix_doc_uk_marc21

This routine converts UKMARC to USMARC.

fix_doc_uni_100

Automatically adds date of publication to positions 09-12 (or corrects the existing values) of the UNIMARC 100 field according to the date entered in UNIMARC 210 field, subfield \$d.

fix_doc_uni_100_advanced

The `fix_doc_uni_100_advanced` program is similar to the `fix_doc_uni_100` program. This program automatically inserts or corrects the dates in the 09-12 and 13-16 positions of the UNIMARC 100 field according to the date entered in the UNIMARC 210 field, subfield \$d. Additionally, the dates of the 210 field (subfield \$d) are standardized. For example, for dates like 198? or 19?, the fix program replaces question marks and spaces with "-" (hyphens).

fix_doc_usm_001

Automatically creates a 001 field. The value is taken from the sequence "last-001-number" in UTIL G/2. If the 001 field already exists, a new 001 field is created based on the "last-001-number" and the old field is stored in a new 035 field. In addition, note that if the 003 field is also present, then the program deletes the field and adds its contents to subfield \$b of the newly created 035 field.

fix_doc_usm_222

This program adds a new 222 MARC 21 field to SE (serial) format records. The 222 field is created from the 245 field, subfields \$a and \$b.

fix_doc_usnaf

Adds to the cataloging record 001\$\$a and 010 \$\$a fields with a USNAF control number and prefix, (for example, 001\$\$aABC1001 / 010 \$\$aABC1001). The values (last USNAF number and prefix) are taken from the *last-usnaf-number* sequence in UTIL G/2.

fix_doc_japanese

This routine is used to communicate with an external product used in the Japanese market (Happiness) to enrich a document with segmented and transliterated versions of the data. The URL address of the external product should be defined as a parameter in column 3 of `tab_fix`.

Note that fields returned from the external product longer than the maximum number of characters allowed are split into several separate smaller fields. This should be viewed and edited by the cataloger.

fix_doc_zero_ldr_00_04

This program is used to set to zeros the first five character positions of the LDR field (00-04). These positions contain a numeric string that specifies the length of the entire record. The number is right-justified and each unused position contains a zero. The LDR of the records in the system usually contains either spaces - when the record is created through the Cataloging module - or the original logical record length from the imported record. In both cases, these values are incorrect and misleading. This fix should be used during conversion, import and cataloging.

Note that ALEPH's export routines calculate the record's length automatically when the total record is assembled for exchange.

expand_doc_fix_abbreviation

This program can be used both as an expand program and as a fix program. Refer to the Expand Record section in the Indexing module for more details (expand_doc_fix_abbreviation).

expand_doc_type

This program can be used both as an expand program and as a fix program. Refer to the Expand Record in the Indexing module for more details (expand_doc_type).

fix_doc_create_7xx_kor and fix_doc_create_7xx_marc

The following two fix routines create full 76X-78X fields based on the existing slim 76X-78X fields, that is: 76X-78X fields with subfield \$\$w data.

- fix_doc_create_7xx_kor is used for KORMARC BIB record
- fix_doc_create_7xx_marc is used for MARC21 BIB record.

The cataloger should delete the LKR field before updating the \$\$w subfield in the 76X-78X and recreate the LKR manually.

The structure and content of the created full 76X-78X linking fields are as follows:

KORMARC (fix_doc_create_7xx_kor)

The order of the 76X-78X subfields is as follows:

- \$\$t (title)
- \$\$s (uniform title)
- \$\$a(author)
- \$\$b(edition)
- \$\$d(publisher)
- \$\$r(report number)
- \$\$u(STRN)
- \$\$x(ISSN)
- \$\$y(CODEN)
- \$\$z(ISBN)
- \$\$w(system number)

The source information for every subfield is taken from the target record in the BIB library.

7xx subfield	Source information to be taken from the target record
\$\$t (title)	245 \$\$a, \$\$f, \$\$g, \$\$k, \$\$n, \$\$p
\$\$s (uniform title)	130 \$\$a or 240\$\$a If there is no 130, use 240; if there is no 240, do not create \$\$s
\$\$a (author)	100 tag (all subfields) -> 110 tag (all subfields) -> 111 tag (all subfields) -> 700 tag (all subfields) -> 710 tag (all subfields) -> 711 tag (all subfields) The system looks first for 100, then for 110, and then 111, and so on, until it finds a matching field. The first field found is used.
\$\$b (edition)	250 \$\$a
\$\$d (publisher)	260 \$\$a, \$\$b, \$\$c
\$\$r (report number)	088 \$\$a
\$\$u (STRN)	027 \$\$a
\$\$x (ISSN)	022 \$\$a
\$\$y (CODEN)	030 \$\$a
\$\$z (ISBN)	020 \$\$a
\$\$w (BIB library code) + target record system number, entered by the cataloger	Remains as it was filled by the cataloger.

Note:

Data from input fields that have multiple subfields are created without the input subfields.

The following fields are mapped in the fix_doc routine:

- 760
- 762
- 765
- 767
- 770
- 772
- 773
- 774
- 775
- 776
- 777
- 780

- 785
- 786
- 787

with the following subfields:

- \$\$t(title)
- \$\$a(author)
- \$\$b(edition)
- \$\$d(publisher)
- \$\$x(ISSN)
- \$\$z(ISBN)
- \$\$w(system number)

MARC21 (fix_doc_create_7xx_marc)

The order of the 7xx subfields should be as follows:

- \$\$a (author)
- \$\$t (title)
- \$\$s (uniform title)
- \$\$b (edition)
- \$\$d (publisher)
- \$\$r (report number)
- \$\$u (STRN)
- \$\$x (ISSN)
- \$\$y (CODEN)
- \$\$z (ISBN)
- \$\$w (system number)

Source information for every subfield should be taken from the target record (\$\$w) in the BIB library (prefix of \$\$w)

7xx subfields	Source information to be taken from the target record
\$\$a (author)	100 tag (all subfields) -> 110 tag (all subfields) -> 111 tag (all subfields) -> 130 tag (all subfields) -> The system will first look for 100 and then for 110 and then 111, and so on, until it finds a matching field. We should derive it from the first field.
\$\$t (title)	245 \$\$a, \$\$f, \$\$g, \$\$k, \$\$n, \$\$p
\$\$s (uniform title)	240 \$\$a the first tag
\$\$b (edition)	250 \$\$a
\$\$d (publisher)	260 \$\$a, \$\$b, \$\$c
\$\$r (report number)	088 \$\$a
\$\$u (STRN)	027 \$\$a

\$\$x (ISSN)	022 \$\$a
\$\$y (CODEN)	030 \$\$a
\$\$z (ISBN)	020 \$\$a
\$\$w (BIB library code) + target record system number, entered by the cataloger	Remains as it was filled by the cataloger.

The subfields allocation is same as in above KORMARC information.

fix_doc_create_lkr

Create LKR fields depending on the 76X-78X fields \$\$w data.

In order to create a functional LKR field, the slim 76X-78X field must have data in \$\$w that can be used to create LKR \$\$b and \$\$l.

The structure of the LKR field is:

- \$\$a - link type (UP, PAR, DN) – see the following section 1.
- \$\$b - system number of the target record – see the following section 3.
- \$\$l - target BIB library code – see the following section 3
- \$\$r- MARC Tag and Indicators – see the following section 4
- \$\$m - content of 245 \$\$a of the same record
- \$\$n - content of 245 \$\$a of the target record

LKR fields will be created only when the link type is PAR, UP or DN.

NOTE: subfields \$\$m and \$\$n will be generated by the system using fix_doc_lkr_up routine. The procedure should be set in the library's tab_fix table in the INS2 section.

1. Setting type of link (LKR \$\$a)

The 76X-78X field tag determines the link type:

- 760 – UP
- 762 - DN
- 765 - PAR
- 767 – PAR
- 770 - DN
- 772 - UP
- 773 – UP
- 774 - DN
- 775 - PAR
- 776 - PAR
- 777 - PAR
- 780 - PAR
- 785 - PAR
- 786 - PAR
- 787 – PAR

2. Setting the document system number(\$\$b) and BIB library code(\$\$l)

The cataloger enters

- 78002 \$\$w(YUL02)48923

\$\$b should be taken from \$\$w of 76X-78X with deleting the prefix

- \$\$b48923

\$\$l should be taken from the prefix of \$\$w of 76X-78X with deleting the parentheses.

- \$\$lYUL02

If there is no prefix in the content of \$\$w of 76X-78X, the same library code as the source record could be set to \$\$l.

3. Setting the \$\$r

3 digits of MARC Tag and 2 digits of Indicators from slim 76X-78X field created by the cataloger will be used.

- \$\$r78002

fix_doc_tag_008_heb

This fix routine gets the years in Hebrew letters or in Arabic numbers, (entered in MARC 21 260 field, subfield \$c. If 260\$\$c is missing, 264\$\$c is applied) converts them to Latin numbers, and places them in field 008, positions 7-10. If 264\$\$c is applied, it is done based on the following priority order: 2nd indicator 1,0,3,7.

fix_doc_signatura

This fix routine gives a unique inventory number to a specific catalogue entry. It figures a new inventory number for a given year and series and places it in 090 field of the document. In general, every year starts from 1. See the following example:

2009 a 12.

- 2009 – represents the year 2009
- a – the code for a series
- 12 – the assigned running number that is derived from the system counter util g/2 - last-signa-a.

You can define a separated counter per series code, for example, last-signa-a, last-signa-b, last-signa-c, etc.

fix_doc_aut_008_pos_29

Fix routine for the Authority record. Parameters must be defined in column 3 of tab_fix. The parameters list the MARC fields, separated by a comma, that are checked by the program. Up to 25 MARC fields can be listed.

Sample from ./usm10/tab/tab_fix

```
! 1          2          3
```

```
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
```

```
INS fix_doc_aut_008_pos_29      400,410,411,430,451,500,510,511,530,551
```

The program performs the following:

1. Sets 008/29 to “n” if the record does not contain any of the MARC fields listed in column 3.

2. Sets 008/29 to “a” if the record does contain any of the MARC fields listed in column 3.

If no parameters are defined, fix_doc_aut_008_pos_29 is inactive.

If 008/29 is “b” to begin with, none of the above is performed.

fix_doc_aut_008_pos_32

Fix routine for the Authority record. No parameters are defined in column 3. The program performs the following:

1. Sets 008/32 to “a” if the record contains a 100 field, unless it also contains a 670 field whose \$a subfield begins with a left square bracket – 670 \$a [
2. Sets 008/32 to “b” if the record contains both a 100 field, and a 670 field whose \$a subfield begins with a left square bracket – 670 \$a [
3. Sets 008/32 to “n” if the record does not contain a 100 field.

12.2 fix_doc.lng

The fix_doc.lng table enables you to define the menu options that are displayed when the cataloger chooses the Fix Record function or the Derive New Record function from the Edit menu of the Cataloging module. The following is a sample of the fix_doc.lng table:

```

! 1      2 3              4
!!!!-!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
INS     N L Perform fixes as executed when document is updated (INS)
PUNC    N L Fix document's punctuation
04-01   N L Convert UNIMARC records to MARC 21 records
008     N L Update 008 field from 260 field
AUT     Y L Create authority record from current bibliographic record
ANA     C L Create an analytic record from the current record

```

Key to the fix_doc.lng table:

Column 1 - Procedure ID

This is the unique code by which the system identifies the procedure. It must be an ID defined in column 1 of the tab_fix table.

Column 2 - Fix Current Record/Derive New Record

This column defines whether a new record is going to be created when performing a fix routine, or if the current record is going to be fixed. The possible values are:

Y - Open as new record (unconditionally).

C - Open as a new record (only if the current record has been previously saved on the server).

N - Fix current record.

Routines that have been set to N appear under the Fix record option in the Edit menu. Routines that have been set to Y or C appear under the Derive New Record option in the Edit menu. Note that if the routine is set to C and the current record for the Derive routine is a local record that has not yet been saved on the server, then this routine is not displayed in the Fix Procedure window.

Column 3 - ALPHA

ALPHA code. Must always be L.

Column 4 - Text of menu option

Enter a description of the procedure up to 45 characters in length. This text will appear under the Fix record option and the Derive new record option in the Edit menu according to the setup of column 2.

It is possible to write external programs for fixing records. External programs can be written in any programming language and can be executed without linkage to ALEPH 500. They are particularly useful for special on-site developments.

The program must reside in \$aleph_exe and it should have no extension. When the program is compiled, it will be placed in \$aleph_proc.

12.3 fix_doc_track

This fix routine is used to store the change history of bibliographic and authority records in the new Z00T Oracle table. The fields that are tracked are listed in the Parameters column (Col. 3). For example:

```
! 1          2          3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
INS2 fix_doc_track          100##,310##,500##,651##
```

Note that if there are multiple occurrences of a field, only the first one is tracked. In the above example, if the record has a 10010 field and a 10011 field, then only the one that is found first in the document will be tracked.

The change history may be viewed in the Cataloging GUI by selecting the **Edit\View Records History** menu option.

13 Locate Function

The Locate function of the Cataloging module enables you to find records in other databases or in your local database that are similar to the one currently being edited. The System Librarian is responsible for setting up the criteria that the system uses in order to determine which records are similar (for example, you can decide that if the records have the same words in the title and author fields, then the records are "similar"). You can define the criteria by editing the `tab_locate` table located in the library's tab directory.

Note that the criteria defined in this table also affect the Locate function in the Search function.

The `tab_locate` table defines the locate routine that is to be used when searching for a record in other databases. Multiple lines can be set up for one library, in which case

all lines are taken with an AND condition between them. The `tab_locate` table should include both the source and target library.

Following is a sample of the `tab_locate` table:

1	2	3	4	5	6
!!!!!!!!!!!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!					
USM01	100## ab		wau=	locate_str_3	3
USM01	245## -c		wti=	locate_str_0	90
USM01	008##		wyr=	locate_str_2	
UNI01	100## a		wau=	locate_str_1	
UNI01	245## a		wti=	locate_str_0	

Key to the `tab_locate` table:

Column 1 - Library Code

Enter the library code of the library in which you want to locate records.

Column 2 - Tag and indicators

Enter a field tag that is used as a "locate" parameter. You can define specific indicators. Use the # character to indicate any indicator. Note that this is always the local tag (for example, see the tag definitions for locating in UNI01 -UNIMARC type library- from USM01 which is a MARC 21 library).

Column 3 - Subfield

Enter the subfields that will be used to build the locate string. The "-" sign can be used to mean "all subfields except for". For example, if this column is configured with: -ab, then all subfields except for 'a' and 'b' will be used.

Column 4 - Find command

Enter the WRD code that is used with the find command to search for similar records.

Column 5 - Extract function

The extract functions define how the contents of the field are going to be treated.

Extract functions:

locate_str_0:

Takes subfield content as is.

locate_str_1:

Runs "build_filing_key" on a subfield and takes the 2 longest words. A word must be at least two characters in order to be considered a "word". If the subfield has only one word, that word is taken.

locate_str_2:

Takes the year from the 008 field (position 8, length 4).

locate_str_3:

Works similarly to `locate_str_1`, but takes the number of longest words specified in the Column 6 parameters (for example, the three longest words).

locate_str_sys_no:

Uses a doc number in a specified field to perform an exact match.

Column 6 - Parameters

Enter the parameters that will be used by the extract function defined in column 5. Such parameters can be the number of words that will be used by locate_str_3 or the breaking procedure that will be used by locate_str_0 or locate_str_1.

Bases for the locate function are defined in the ALEPHCOM/TAB/LOCATE.DAT file. This file affects both the Locate function in the Cataloging module and the Locate function in the ILL module. Note that you can define a separate file for the Cataloging module. You do this by adding a locate.dat file to the catalog directory of the library (./pc_tab/catalog). This file must be in the same format of the locate.dat of the alephcom directory. Note that if the file is added, even if it is empty, then the bases in the Locate window of the Cataloging module are taken from the locate.dat file of the catalog directory. If the file is left empty, then no bases are displayed from the Cataloging module even if bases have been defined for the locate.dat file in the alephcom directory.

The "locate" section in the CATALOG/TAB/CATALOG.INI file defines whether or not the record found using the Locate function should be merged automatically with the current record.

14 Duplicate Record Function

The Duplicate Record function enables you to copy the currently displayed record and then edit the copy. The new record is located on your local drive.

It is up to you, the System Librarian, to determine whether the new record should be assigned automatically to the Home Library (the library to which the user is currently connected), assigned automatically to another specific library, or assigned to the library of the cataloger's choice (in which case, a list of all available libraries is displayed for the cataloger to choose from).

In order to determine which of the above is in effect, open the CATALOG.INI file (found in the client's CATALOG/TAB directory). Go to the [DuplicateRecord] section. Following is an example of what you can find there:

```
[DuplicateRecord]
Library=HOME
```

If you want the new record to be assigned automatically to the Home library, type *HOME* to the right of the equal (=) sign. If you want a different library, type the code for the library, for example, *USM01*. If you want the cataloger to choose from a list of all available libraries (that is, all libraries listed in the per_lib.ini file in the CATALOG/TAB directory), type *ALL*. If you want to define the list of libraries that the cataloger can choose from, type the list of libraries. For example:

```
Library=USM01,USM20,ACC01,UBW01
```

15 Importing Updated Tables

You can determine whether or not the system automatically imports updated Catalog tables when the Cataloger opens the Cataloging module. To determine this, go to the ALEPHCOM/TAB directory and open the ALEPHCOM.INI file. Go to the section labeled [Package]. Following is an example of the relevant section:

```
[Package]
AlwaysImportFiles=Y
```

Enter *Y* to the right of the equal sign if you want the system to import updated tables automatically.

Enter *N* to the right of the equal sign if you want the system to ask the Cataloger whether or not he wants to import the updated tables.

Note that this section also determines whether or not the updated printing templates package is automatically downloaded to the client when connecting to any of the modules.

16 Floating Keyboard

The Floating Keyboard enables you to insert characters that are not present in your workstation's standard keyboard. The Floating Keyboard is configured by the System Librarian according to the needs of the library. Following is an example of a floating keyboard.

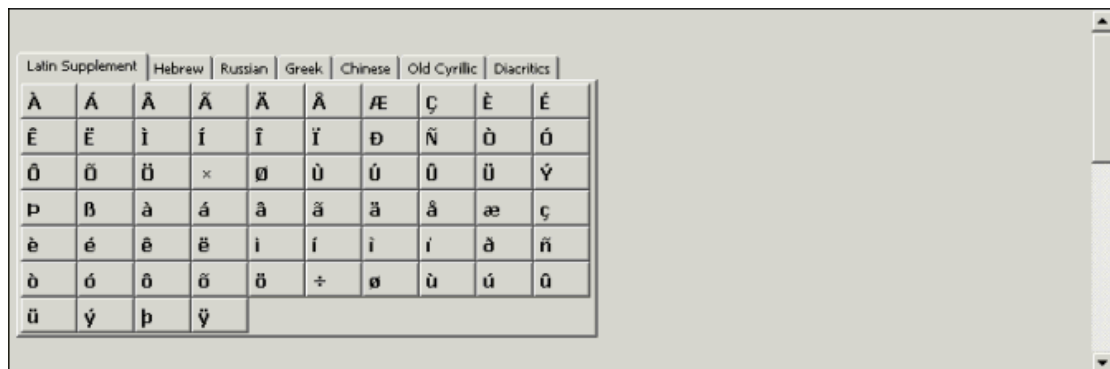
Three files define the Floating Keyboard setup:

- Keyboard.ini
- Keyboard.txt
- Font.ini

All files are located in the ALEPHCOM/TAB directory.

Keyboard.ini

The keyboard.ini defines the configuration settings.



The screenshot shows a floating keyboard window with a grid of characters. The window has a title bar and a scroll bar on the right. The grid is organized into columns labeled 'Latin Supplement', 'Hebrew', 'Russian', 'Greek', 'Chinese', 'Old Cyrillic', and 'Diacritics'. The characters are arranged in rows, with some cells containing multiple characters or symbols.

Latin Supplement	Hebrew	Russian	Greek	Chinese	Old Cyrillic	Diacritics			
À	Á	Ā	Ā	Ā	Æ	Ç	È	É	
Ê	Ë	Ī	Í	Ī	Ī	Ð	Ñ	Ò	Ó
Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý
Ɔ	Ɔ	à	á	â	ã	ä	å	æ	ç
è	é	ê	ë	ì	í	î	ï	ð	ñ
ò	ó	ô	õ	ö	÷	ø	ù	ú	û
ü	ý	þ	ÿ						

The sample below matches the example of a Floating Keyboard shown above.

```
[Main]
Title=Keyboard

[WindowLocation]

KeyboardWindowPosition=189,267
KeyboardWindowRelocate=Y

[Tabs]
NoTabs=7

[Tab1]
Caption=Latin Supplement
NoCols=10
BtnWidth=40
BtnHeight=25

[Tab2]
Caption=Hebrew
NoCols=10
BtnWidth=40
BtnHeight=25

[Tab3]
Caption=Russian
NoCols=10
BtnWidth=40
BtnHeight=25

[Tab4]
Caption=Greek
NoCols=10
BtnWidth=40
BtnHeight=25

[Tab5]
Caption=Chinese
NoCols=10
BtnWidth=40
BtnHeight=25

[Tab6]
Caption=Old Cyrillic
NoCols=10
BtnWidth=40
BtnHeight=25

[Tab7]
Caption=Diacritics
NoCols=10
BtnWidth=40
BtnHeight=25
```

Table sections:

[WindowLocation]

This section defines the position of the Floating Keyboard and whether or not it is possible to relocate it. Note that in the Cataloging module, this option is not in use. In the Cataloging module, the keyboard is displayed in the lower pane.

[Tabs]

This section defines the number of tabs that appear in the Floating Keyboard.

[Tab(number)]

For example, [Tab3]

This section defines the configuration settings for each tab of the keyboard.

Caption: Defines the caption of the tab (for example, Russian).

NoCols: Defines the number of columns for the tab.

BtnWidth: Defines the width of the character keys for the tab.

BtnHeight: Defines the height of the character keys for the tab.

Keyboard.txt

The Keyboard.txt file defines the characters that are displayed in each tab.

The sample below matches the example of a keyboard shown above.

```
! Unicode code
!!!!!!!!!!!!!!

[Latin Supplement]
!!!!!!!!!!!!!!
\00C0
\00C1
\00C2
\00C3

etc...

[Hebrew]
!!!!!!!!!!!!!!
\05D0
\05D1
\05D2
\05D3

etc...

[Russian]
!!!!!!!!!!!!!!
\0410
\0411
\0412
\0413

etc...

[Greek]
!!!!!!!!!!!!!!
```



```
\0386
\0388
\0389
\038A
```

etc...

[Chinese]

!!!!!!!!!!!!

```
\4E10
\4E11
\4E12
\4E13
```

etc...

[Old Cyrillic]

!!!!!!!!!!!!

```
\0410
\0411
\0412
\0413
```

etc...

[Diacritics]

!!!!!!!!!!!!

```
\02BB
\0307
\0324
\0310
```

etc...

This file contains one column. This column contains the Unicode value of the character that is inserted in the cataloging draft.

Note that the table is divided according to the tabs for the keyboard. Each section should be entered in the same order in which it is defined in the Keyboard.ini file. The link between a tab in the two files is determined by order and not by the capture.

Font.ini

The Font.ini file contains the font definitions.

Note that is possible to define different fonts for different Unicode ranges (columns 2 and 3 of the file).

The following is an example of the Font.ini file for the floating keyboard:

!	1	2	3	4	5	6	7	8
9	!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!-!!!!-!!-!-!-!-!							
!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!								
AlephKeyboard		0000	00FF	Tahoma			Y	N
16	DEFAULT_CHARSET							

AlephKeyboard	0401 045F	Tahoma	Y	N	N
16 DEFAULT_CHARSET					
AlephKeyboard	0384 03CE	Tahoma	Y	N	N
16 DEFAULT_CHARSET					
AlephKeyboard	05D0 05EA	Tahoma	Y	N	N
16 DEFAULT_CHARSET					
AlephKeyboard	0000 FFFF	Bitstream Cyberbit	Y	N	N
16 DEFAULT_CHARSET					

For more information on the Font.ini file refer to the Font Definitions (Font.ini file) section of the General chapter.

17 Authorizations

17.1 Allowed and Denied Tags

The `permission.dat` table, located in the library's `pc_tab/catalog` directory, defines allowed and denied tags for different catalogers.

Following is a sample of the table:

```

!1          2          3 4
!!!!!!!!!!-!!!!!!-!-!!!!!!!!!!!!
YOHANAN    ##### Y
YOHANAN    650## N
OMRI       ##### Y
OMRI       100## N
TAMI       ##### Y
TAMI       245## N
YIFAT      ##### N

```

Key to `permission.dat`:

Column 1 - User Name

This is the unique string by which the system identifies the cataloger/user.

Column 2 - Tag Code

This column contains the allowed or denied tag and indicators. Use the hash (##) as a placeholder for undefined tags and/or indicators (for example, 100## means tag 100 any indicators; ##### means ALL tags).

Column 3 - Type of Permission

Values are Y and N. Y is used for allowed tags and N for denied tags. In the above sample of the table, the user OMRI is authorized to edit all fields except for the 100 field.

In the Cataloging module, denied tags will appear in a different color.

A user that does not have an entry in the `permission.dat` table is denied permission to edit any tag. If the library does not want to use the `permission.dat` mechanism, the table can be removed and all users will then be allowed to edit any tags.

Note that users that have cataloging proxies do not need to be listed in this table. When the cataloging tables are repacked, users of this type are granted the rights assigned to their cataloger proxy.

17.2 Cataloging "OWN" Permissions

The system librarian can assign a group of allowed OWN values for a cataloger. This can be done by setting up the `tab_own` table in the library's `tab` directory.

Up to five different OWN values of cataloging records can be allowed for a single OWN value of a user.

Following is a sample of the table:

!	1	2	3	4	5	6
!!!!!!!-!!!!!!!-!!!!!!!-!!!!!!!-!!!!!!!-!!!!!!!						
CAT	AA	BB				
CAT1	BB	CC	DD			

In this example, any user with the value CAT in its Cat. OWN Permission field has authorization for updating records with OWN values of AA and BB. Those users with the value CAT1 in their Cat. OWN Permission field have authorization for updating records with OWN values of BB, CC and DD.

Note that it is possible to assign more than 5 different OWN values of cataloging records to a user's OWN value by using the hash (#) character as a wildcard.

Following is a sample of the table in which the # sign is used to cover more OWN values:

!	1	2	3	4	5	6
!!!!!!!-!!!!!!!-!!!!!!!-!!!!!!!-!!!!!!!-!!!!!!!						
MECAT	ME###					
MECAT1	#####					
MECAT2	#####					
MECAT3	ME###	GR####				
MACA##	MACAT1					

Based on the sample above:

ME### includes, for example, MEDUC, MELEC, and so on.

includes all OWN values that are up to five characters.

includes all possible OWN values (this is equal to the GLOBAL authorization).

Note

If a User's OWN value needs to be assigned more than five different record's own values, (without using the hash (#) character as a wildcard), you can define multiple lines for the same user's OWN. For example:

!	1	2	3	4	5	6
!!!!!!!-!!!!!!!-!!!!!!!-!!!!!!!-!!!!!!!-!!!!!!!						
! CAT	MED	HYL	HIL	LAM	LAW	

! CAT	LIT	MUS	WID	HILR	BCU
-------	-----	-----	-----	------	-----

Key to the tab_own table:

Column 1 - User's OWN Permission

This column contains the value of the Cat. OWN Permission field assigned to the user(s). Use the hash (#) character as a placeholder for any character. For example, CAT## includes users with OWN Permission CAT, CAT1, CATXX, and so on.

Columns 2 to 6 - Record's OWN value

Columns 2 to 6 contain the record's OWN values which the user with the OWN permission defined in column 1 is allowed to update. Use the hash (#) character as a placeholder for any character. For example, ME### includes, MEDUC, MELEC, and so on.

If a catalog proxy is assigned to the user (see the Staff Privileges - User Information - Password section, then the OWN values for the user are taken from the proxy's record.

17.3 Holdings Filter

The holdings records displayed in the Cataloging navigation tree and in the *HOL records* tab in the Cataloging module lower pane can be filtered based on the holdings record's OWN field. Only users with an OWN Permission value that is the same as the value in the OWN field in the holdings record will be able to see it.

This filter is dependent on the tab_own table. In addition, the OWN-FILTER value in tab100 must be Y.

18 Merging Records

The fix_doc_merge program is used to merge or overlay cataloging records according to the merging routines defined in the tab_merge table located in the library's tab directory. Column 3 of the tab_fix table is used to define the merging routine that matches the relevant section in the tab_merge table.

The following is a sample of an entry for the fix_doc_merge program in the tab_fix table:

```
! 1                               2                               3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!->
FIX  fix_doc_merge                OVERLAY-01
```

The "OVERLAY-01" routine must match a routine in the tab_merge table. The following is a sample of the tab_merge table:

```
! 1                               2                               3
!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!->
OVERLAY-01 merge_doc_overlay      01
OVERLAY-02 merge_doc_overlay      02
OVERLAY-03 merge_doc_replace
```

Key to the tab_merge table:

Column 1 - Routine Name

This column is used to define the merging routine. It matches the routines defined in column 3 of the tab_fix table.

Column 2 - Merging Program

This column contains the merging program. The following are the available options:

merge_doc_replace:

This program replaces the contents of the original record with the contents of the new record, retaining the CAT fields from both records.

merge_doc_overlay:

This program merges/overlays the record according to the overlay specifications defined in the tab_merge_overlay table of the library's tab directory.

merge_doc_adv_overlay:

This program merges/overlays the record according to the overlay specification defined in the tab_merge_adv_overlay table of the library's tab directory. This table shares the same purpose as the tab_merge_overlay table and acts in a similar manner, with an added level of complexity. The additional functionality is based on determining which record is "preferred" when the merge is performed. When merge_doc_adv_overlay is chosen from tab_merge, the system first consults tab_preferred to set the "preferred" program that will be used and the accompanying "weights" table that is used to evaluate the two records. This program is usually used when loading records into the system.

Column 3 - Merge Set

This column contains the merge set to be applied when the merge_doc_overlay or the merge_doc_adv_overlay programs are performed. The merge set must match a merge set defined in the tab_merge_overlay/tab_merge_adv_overlay tables of the library's tab directory.

Note that when the overlay programs are used, the system librarian is in charge of defining which fields are retained or overwritten when merging/overlapping two cataloging records. This is done by editing the library's tab_merge_overlay/tab_merge_adv_overlay tables located in the library's tab directory.

The merge_doc_overlay function runs when the Paste record option is selected from the Edit menu of the Cataloging module. The system uses the definitions of the tab_merge_overlay table if the following line is defined in the tab_fix table of the library's tab directory:

```
MERGE fix_doc_merge          (routine name for tab_merge)
```

The merge_doc_overlay function can also be used when the Locate Similar Records option is selected from the Edit menu of the Cataloging module. The system uses the definitions of the tab_merge_overlay table if the following line is defined in the tab_fix table of the library's tab directory:

```
LOCAT fix_doc_merge          (routine name for tab_merge)
```

Following is a sample of the `tab_merge_overlay` table:

```
!1 2 3 4
!!-!-!-!!!!
01 1 Y #####
01 1 N 008##
01 1 C 245##

01 1 Y 245##
```

Key to the `tab_merge_overlay` table:

Column 1 - Merge set

This column is used to define different merging routine sets. Up to 99 different merging routine sets can be defined; the values are 01 to 99. Note that the routine must match the definitions in the `tab_merge` table of the library's `tab` directory. For example, if you want to work with the 03 routine, then the relevant `fix_doc_merge` section of the `tab_fix` table must be attached to the routine that in the `tab_merge` table is set to work with the merge set 03. Additionally, note that the lines of the table are limited to 99.

Column 2 - Merging direction

Values are 1 and 2. 1 defines lines for the original record, that is, the document into which fields are merged/pasted. 2 defines lines for the document from which fields are copied.

Column 3 - Action

Values are Y, N and C:

Y - For the original record (1) - retains the field.

For the copied record (2) - copies the field.

N - Does not retain the field.

C - Retains the field only if it does not appear in the other record.

Column 4 - Tag code

This column contains the field tag and its indicators. Use the hash (#) as a placeholder for undefined tags and/or indicators (for example, 100## means tag 100 any indicators; ##### means ALL tags).

This column can also be used for subfield and subfield contents to use as filters, as shown in the following example:

```
01 2 Y 590##5,*abc*
```

In the above example, the tag 590 is disregarded if subfield \$5 of the field does not contain the string "abc" as part of its contents.

In the example above, all fields are taken from the original document (1), except the 008 field. The 245 field is always taken from the copied record. If the copied record does not have a 245 field, the 245 field of the original record is retained. Otherwise it is overlaid from the second document to the original record.

Note that the search for the code is sequential. For example:

```
01 1 N 008##
01 1 Y #####
```

At first, the system will not take the 008 field because of the N in column 3 for the field. Then, the system continues "reading" the next line that defines that all fields should be taken. The result is that the 008 field is taken, too.

19 Updating the Tables Package

After making changes to any of the tables of the catalog directory (\$data_root/pc_tab/catalog), the system librarian is in charge of repackaging the cataloging tables. The cataloging tables are repackaged by performing UTIL M/7. This updates the packaged file of tables (pc_cat.pck) in the library's catalog directory. When a user connects to a home library in the Cataloging module, the system compares the tables on the client with the date of the pc_cat.pck package on the server. If the dates are different and the *AlwaysImportFiles* flag in alephcom.ini (under [Package]) is set to "N", then the user is prompted to update the tables on the client. If the dates are different and the *AlwaysImportFiles* flag is set to "Y", then the tables are imported automatically.

Among other tables/files, the catalog directory contains cataloging forms (for example, 008_bk.lng for MARC 21), cataloging templates (for example, 008_bk.lng for MARC 21), help files (in the HTML subdirectory), codes for the FMT field (formats.lng), field contents for fixed text fields (tag_text.dat), list of valid tags and aliases (codes.lng), and so on.

20 Subfield Punctuation

The tab_subfield_punctuation table in the library's tab directory is used to define subfield punctuation for fields. Punctuation for fields is necessary when the system automatically updates the bibliographic record from a linked authority record. When the bibliographic record is updated from the authority database the system always uses the preferred term (1XX) from the authority record. Originally the bibliographic record may have more data than the authority record. This data should be retained. In MARC, authority records do not have end punctuation while bibliographic records do. The tab_subfield_punctuation table is used to add end punctuation to the updated field. The table can be also used to add punctuation between the end of the preferred term from the authority record and the additional subfields retained from the bibliographic record (for example, between subfield \$a - personal name - and subfield \$t - title of MARC 21 600 field). The following is a sample of the tab_subfield_punctuation table:

```
!2  3 4 5      6
!-!!!!!!-!-!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!
A 1#### a    .      .
A 1#### d    .      -.
A 100## a 4  ,
A 100## d 4  ,
A 110## b    .      .
```

Key to tab_subfield_punctuation:

Column 1 - Program code

Use always "A".

Column 2 - Tag and indicators

Contains the field tag with indicators for which subfield punctuation is being defined. Use the hash (#) as a placeholder for undefined indicators.

Column 3 - Subfield code

Enter the subfield to which the end punctuation is going to be added.

Column 4 - Following subfield code

Enter the subfield that follows after the end punctuation added to the subfield defined in the previous column.

Column 5 - Punctuation to add

Enter the punctuation signs that should be added to the subfield.

Column 6 - If punctuation

This column is used to determine whether or not the punctuation defined in column 5 is added to the field. If the field already ends with the punctuation defined in column 6, punctuation from column 5 is not added.

21 Validation of Contents of a Field

You can set up ALEPH to check the contents of some fields. This is done through the `check_doc_line_contents` table in the library's `tab` directory.

Following is a sample of the `check_doc_line_contents` table:

```

! 2      3          4                          5
!!-!!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!
## 020## a isbn
## 022## a issn
## 7#### x issn
## 260## c range           1850 2002
## 260## c number_length   4
## 022## a length          9

```

Key to the `check_doc_line_contents` table:

Column 1 - Record's format

Enter a specific record format (for example, BK), or use ## as a wildcard to indicate that the field is appropriate for any format. Refer to Record Formats on page 9 for more information on record formats.

Column 2 - Field code

Field code to be checked. Use the hash (#) as a placeholder for undefined tags and/or indicators (for example, 020## or 7####).

Column 3 - Subfield code

Enter the subfield code of the subfield to be checked. If the column is left blank, then the field is taken as is.

Column 4 - Name of check program

The existing check programs are **isbn**, **issn**, **issn-isbn**, **length**, **number_length**, and **range**:

isbn - verifies that the ISBN entered in the field is a valid ISBN (including check digit). This routine is capable of validating both types of ISBN: 10-digit ISBNs and 13-digit ISBNs.

Both 10-digit ISBNs and 13-digit ISBNs are considered valid.

issn - verifies that the ISSN entered in the field is a valid ISSN (including check digit).

issn-isbn – Verifies that the field is either a valid ISSN number or a valid ISBN number. ISSN numbers may be missing the hyphen and ISBN numbers may be 10 or 13 digit numbers.

length - verifies that the length of a numeric string matches the values defined in column 5.

number_length - verifies that the number_length of a numeric string matches the values defined in column 5.

range verifies that the numeric string entered in the field matches the range defined in column 5.

Column 5 - Values to check

For **length** enter <length> (for example, for subfield \$c of MARC 21 field 260, the length is 4 for the year).

For **range** enter <from> <to> (for example, for subfield \$c of MARC 21 field 260, enter reasonable values for the range of the year, say, 1850 - 2001).

22 Check Field Occurrences and Dependency between Fields

Definitions for field occurrences and dependency between fields for checking routines are set up in the `check_doc_doc` table in the library's tab directory.

The table contains two sections:

- OC
- D

Following is a sample of the **OC section**:

```
OC BK 5001 00 01 100## 110## 111## 130##
OC XX 5002 01 01 245##
OC BK 5003 01 01 260##
OC SE 5007 01 01 310##
```

This section enables you to define which fields are mandatory and their repeatability.

Key to the OC section of `check_doc_doc`:

Column 1 - Section ID

Enter OC for each line of this section of the table.

Column 2 - Record format

Enter a specific record format (for example, BK), or use XX as a wildcard to indicate that the check is appropriate for any format. Refer to Record Formats on page 9 for more information on record formats.

Column 3 - Error message code

Enter the code of the error message that is displayed in the Cataloging module. The code should match the definitions of the check_doc.lng table located in the library's tab directory.

Column 4 - Minimum of occurrences

00 indicates that the field is not mandatory. 01 indicates that the field must be present.

Column 5 - Maximum number of occurrences

If the field is not repeatable, enter 01. If the field is repeatable, you can use values 02 to 99 to define that the field can be repeated up to a particular number of times according to the selected value.

Column 6 - Field code

Enter the field code of the fields for which occurrences are being defined. Up to 5 field codes can be entered (with "OR" implied).

In the above sample of the table, the first line indicates that a record can have only one occurrence of either MARC 21 field 100, or 110, or 111 or 130. These fields are not mandatory.

Note that repeated fields with the same subfield \$6 (this subfield links fields that are different script representations of each other) are considered a single occurrence of the field. This avoids incorrect repeatability messages.

Based on the above sample, if the record contains two occurrences of MARC 21 field 100, as follows:



```
Leader      LDR      -- 01142cam^^2200301Mi^45@0
Control No. 001      -- ocm34447079^
Control No. ID 003      -- OCoLC
Fixed Data  008      -- 941012s19461946cc^^^^^^r^^^^^000^0^chi^d
Personal Name 100 1 6 01
              a Ai, Qing,
              d 1910-
Personal Name 100 1 6 01
              a 艾青,
              d 1910-
Main Title   245 10 6 02
              a Pei fang /
              c Ai Qing [zhu].
```

then no error message is displayed when checking the record; the system considers both 100 fields as a single occurrence.

Following is a sample of the **D** section:

```
D BK 7003 2450# Y 1#### N
D BK 7004 2451# Y 1#### Y
```

This section of the table enables you to define dependencies between fields, such as if one is present another must be present, or if one is present another must not be present.

Key to the D section of `check_doc_doc`:

Column 1 - Section ID

Enter D for each line of this section of the table.

Column 2 - Record format

Enter a specific record format (for example, BK), or use XX as a wildcard to indicate that the check is appropriate for any format. Refer to Record Formats on page 9 for more information on record formats.

Column 3 - Error message code

Enter the code of the error message that is displayed in the Cataloging module. The code should match the definitions of the `check_doc_lng` table located in the library's tab directory.

Column 4 - Field code

Field code for the first part of the condition. Use the hash (#) as a placeholder for undefined tags and/or indicators (for example, 100## or 1#0##)

Column 5 - Type of dependency

This column defines whether the check relates to the field being present or not. Values are Y and N. Use Y to define that the field is present. Use N to define that the field is not present.

Column 6 - Field code

Field code for the second part of the condition. Use the hash (#) as a placeholder for undefined tags and/or indicators (for example, 100## or 1#0##).

Column 7 - Type of dependency

This column defines whether or not the check relates to the field being present. Values are Y and N. Use Y to define that the field is present. Use N to define that the field is not present.

In the above sample of the section, if a record has a 245 field with 0 as first indicator, then the 1XX fields must not be present. If the record has a 245 field with 1 as first indicator, then a 1XX field must be present.

23 Forbidden Errors and Triggers

The `check_doc_mandatory` table can be used to define whether error messages produced by cataloging check routines should activate a trigger or be defined as forbidden.

A cataloging error defined as forbidden does not allow the user to save/update the record, while errors that activate triggers allow database update. The triggers are automatically assigned CAT as the department in the Trigger Department field. Triggers can later be retrieved using the Triggers node on the Cataloging tab; the triggers of a specific record can be retrieved through the **Record's Triggers** command from the **Edit** menu of the Cataloging module.

For example, if the `check_doc_doc` table is used to define that a MARC 21 record must have a 245 field (for example, OC XX 5002 01 01 245##), then you can set error message 5002 to activate a trigger or to be defined as forbidden.

In addition, through the check type (column - col.1), you can define that the error message activates a trigger or is forbidden only in particular instances of the system,

such as, for example, when records are updated or created from the Cataloging module.

Following is a sample of the `check_doc_mandatory` table:

1	2	3	4
!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!-!-			
!!->			
CATALOG-DELETE	0011	M	ADM record points to current document.
	0012	M	HOL record points to current document.
	0013	M	BIB record points to current document.
CATALOG-INSERT	0101	T	Field is a duplicate entry in the INDEX file.
CATALOG-INSERT	0110	T	Field is a new heading in the index list.
CATALOG-INSERT	0161	T	ISBN is incorrect.
CATALOG-INSERT	0162	T	ISSN is incorrect.
	5001	M	A record cannot have more than 1 main entry (1XX).
	5002	M	Required 245 field is either missing or duplicated.
	5008	M	Required 008 field is either missing or duplicated.
	9999	T	Too many errors (must be less than 40).

Key to the `check_doc_mandatory` table:

Column 1 - Check type

The check type defines when the check program is performed. Check programs are assigned to check types in the `check_doc` table of the library's tab directory. The following are the reserved check types:

CATALOG-INSERT: performed when the cataloging record is saved, updated or when the Check Record option is selected from the Cataloging module.

CATALOG-DELETE: performed when the Delete Record from Server option is selected from the Cataloging module.

BATCH-DELETE: performed when the Delete Bibliographic Records (p-manage-33) batch process is run.

NAV-MAP-DELETE: Check programs attached to the NAV-MAP-DELETE check type are run when the Total Delete option is selected from the Record Manager in the Cataloging module.

Z39-INSERT: performed when a record is inserted via Z39.50 ES Update.

Z39-REPLACE: performed when a record is replaced via Z39.50 ES Update.

Z39-DELETE: performed when a record is deleted via Z39.50 ES Update.

Note that if this column is left blank, then the error code defined in column 3, applies for all check types.

Column 2 - Identifying number of the check program

Enter the error code of the check program. User-defined error codes are defined in the `check_doc.lng` table in the library's tab directory. System-defined error codes are defined in the `check_doc` table in the `$aleph_root/error_lng` directory.

Column 3 - Type of error

This column is used to define the type of error. Values are M and T. Errors of type M are considered forbidden errors and do not allow the user to update the record. Errors of type T activate a trigger and allow database update. The record's triggers can be retrieved through the Record's triggers option from the Edit menu of the Cataloging module.

Column 4 - Error message

Optional free-text column. It is non-functional, for information only.

Note that if error code 9999 (Too many errors) is not defined in the table, it is considered by the system as a forbidden error (type 'M').

24 Checking Routines for New Headings in the Headings List

The system librarian can define which fields are ignored for purposes of the check message that informs the cataloger that the heading is a new heading in the headings list (acc file). This is done by including the field in the `check_doc_new_acc` table in the library's tab directory.

Following is a sample of the `check_doc_new_acc` table:

```
! 1
!!!!
245##
260##
```

Key to the `check_doc_new_acc` table:

Column 1 - Field code

Enter the field code of the fields that should be ignored while checking for unique headings in the Heading List (ACC index). Use the hash (#) as a placeholder for undefined tags and/or indicators. In the above sample, the title headings and the imprint headings are ignored by the checking routine for new headings in the list.

Note that for the "New headings" check routine to be performed, the `check_doc_new_acc` program should be listed in the `check_doc` table of the library's tab directory. The `check_doc` table lists all the checking programs that are run when the user chooses the "Check Record" function.

25 Checking Routines for New Headings in the Bibliographic and Authority Headings List

The `check_doc_new_acc_aut` table in the library's tab directory defines the fields that should be ignored when checking for new headings in the Headings List of the relevant authority library and in the Headings List of the bibliographic library.

Following is a sample of the `check_doc_new_acc_aut` table:

```
! 1
!!!!
```

245##
260##

Key to the `check_doc_new_acc_aut` table:

Column 1 - Field code

Enter the field code of the fields that should be ignored while checking for unique headings in the Headings List (ACC index) of the bibliographic library and in the Headings List of the relevant authority database. Use the hash (#) as a placeholder for undefined tags and/or indicators. In the above sample, the title headings and the imprint headings are ignored by the checking routine for new headings in the bibliographic list of headings and in the list of headings of the relevant authority library.

Note that for the "New headings" check routine to be performed, the `check_doc_new_acc_aut` program should be listed in the `check_doc` table of the library's tab directory. The `check_doc` table lists all the checking programs that are run when the user chooses the "Check Record" function.

26 Checking Routines for New Direct Indexes (IND)

The system librarian can define which fields are ignored when the system checks whether or not a duplicate record is opened in the Direct (Z11) Index. This is done by including the field to be ignored in the `check_doc_unique_index` table in the library's tab directory.

Following is a sample of the `check_doc_unique_index` table:

```
! 1
!!!!
050##
020##
```

Key to the `check_doc_unique_index` table:

Column 1 - Field code

Enter the field code of the fields that should be ignored while checking for unique headings in the Direct Request Index. Use the hash (#) as a placeholder for undefined tags and/or indicators. In the above sample, the Library of Congress call number and the ISBN are ignored by the checking routine for new headings in the list.

Note that for the "Duplicate Direct Index" check routine to be performed, the `check_doc_unique_index` program should be listed in the `check_doc` table of the library's tab directory. The `check_doc` table lists all the checking programs that are run when the user chooses the "Check Record" function.

27 Locking Records

27.1 Locking Period for Locked Records

Locked records are automatically unlocked after a period defined by the system librarian by defining the setenv doc_lock_period variable in the pc_server_defaults table located in the \$salephe_root directory. The period is defined in seconds. By default, the variable has been set up to lock records for one hour:

```
setenv doc_lock_period          3600
```

27.2 Lock Status Message

When a cataloger locks a record, the phrase "Locked by current user" is displayed in the Cataloging bar, informing the user that the record has been locked. In addition, when a cataloger loads a record locked by another cataloger, the phrase "Locked by another user" is displayed in the Cataloging bar informing the cataloger that the record is locked. This text can be modified by editing the following entries from the pc_cat_c0203file of the \$saleph_root/error_lngdirectory:

```
2002 0000 L [Locked by another user] System No. $3 - Format $4 - $1
($2)
2003 0000 L [Locked by current user] System No. $3 - Format $4 - $1
($2)
```

28 Check Routines for Check Record

The check_doc table is used to define the check programs that are used in the system and in the environment in which these programs are used.

The following is a sample of the table:

```
!          1                               2                               3
!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CATALOG-INSERT      check_doc_new_acc
CATALOG-INSERT      check_doc_new_acc_aut
CATALOG-INSERT      check_doc_unique_index

Z39-INSERT          check_doc_line
Z39-INSERT          check_doc_line_contents

Z39-REPLACE         check_doc_new_acc
Z39-REPLACE         check_doc_new_acc_aut

CATALOG-DELETE      check_doc_delete_lkr
CATALOG-DELETE      check_doc_delete_item
```

Note that up to 100 programs can be defined in the check_doc table.

Key to the check_doc table:

Column 1 - Check Type

Enter the check type that defines when the check program is performed.

Column 2 - Check Program

Enter the check program(s) that should be performed for the specific check type defined in column 1.

Column 3 - Program Arguments

Contains additional information about the programs, such as table names. This column is used to define additional parameters for the check programs.

28.1 Check Types Available for Column 1 of the check_doc Table:

The following are the available check types:

CATALOG-INSERT: Check programs attached to the CATALOG-INSERT check type are performed when the cataloging record is saved, updated or when the Check Record option is selected from the Cataloging module.

CATALOG-DELETE: Check programs attached to the CATALOG-DELETE check type are performed when the Delete Record from Server option is selected from the Cataloging module.

BATCH-DELETE: Check programs attached to the BATCH-DELETE check type are performed when the Delete Bibliographic Records (p-manage-33) batch process is run.

NAV-MAP-DELETE: Check programs attached to the NAV-MAP-DELETE check type are performed when the Delete Bibliographic record option is selected from the Navigation Window of the Search module.

Z39-INSERT: Check programs attached to the Z39-INSERT check type are performed when a record is inserted via Z39.50 ES Update.

Z39-REPLACE: Check programs attached to the Z39-REPLACE check type are performed when a record is inserted via Z39.50 ES Update.

Z39-DELETE: Check programs attached to the Z39-DELETE check type are performed when a record is inserted via Z39.50 ES Update.

28.2 Check Programs Available for Column 2 of the check_doc Table

The following are the available check programs:

check_doc_852

This program checks whether the sublibrary and collection codes - cataloged in subfields \$b and \$c of the MARC 21 location field (852) of the holdings record - match the definitions of the tab_sub_library and tab40 tables. Note that this program should be included in the check_doc table of the holdings library (xxx60).

check_doc_853

This routine checks if patterns (853/4/5 tags) share the same subfield \$\$8 value.

check_doc_853x

This program checks the presence and the validity of mandatory subfields in the MARC 21 853/4/5 and 853X/4X/5X fields. The check_doc_853x program also

checks dependencies between subfields. For example, if the 853 field has a subfield \$a, then subfield \$a must also be present in the 853X field.

check_doc_adm_lkr

This program checks whether the bibliographic record to which the LKR field in the administrative record is pointing (subfield \$b) is already linked to another administrative record. This program should be used only in administrative libraries (XXX50).

check_doc_aut_008

This check routine produces an error if one of the following is detected in the authority record:

- Position 29 in line 008 has the value 'n', and a 4##### or a 5##### line exists.
- Position 29 in line 008 has the value 'a' or 'b', and no 4##### or 5##### line exists.
- Position 32 in line 008 has the value 'n', and a 1000# or a 1001# line exists.
- Position 32 in line 008 has the value 'a' or 'b', and no 1000# or 1001# line exists.

check_doc_aut_5xx

This program checks whether or not the 5XX field (*See also* from tracing field) cataloged in the authority record has a corresponding entry in the "GEN" index. If there is no matching heading in the "GEN" index, an error message is displayed. If the 5XX field has a corresponding entry, the program also checks whether this entry derives from a 1XX Heading field or from a *See from* tracing field (4XX). If the matching heading derives from a *See from* (4XX), then an error message is displayed.

Note that for the implementation of this program the 5XX fields should be sent to a "GXX" headings index in the authority library. The check_doc_aut_5xx compares the entries in these indexes with the entries in the "GEN" index.

check_doc_aut_duplicate

This program checks whether the authority heading already exists in the GEN index of the authority database. Note that this program should be included in the check_doc table of the authority database (xxx10).

check_doc_doc

This program checks field occurrences and dependencies between fields, according to the definitions of the check_doc_doc table.

check_doc_line

This program checks the validity of indicators and subfields; the presence of mandatory subfields according to the definitions of the check_doc_line table. This program also checks dependencies between subfields.

Checks for fixed fields are also done by this program. To avoid these checks, in column 3 of the check_doc table, place the word -FIXED as seen in the following example:

```
CATALOG-INSERT      check_doc_line      -FIXED
```

check_doc_line_contents:

This program checks the contents of some fields according to the definitions of the `check_doc_line_contents` table.

check_doc_lkr

This program checks the validity of the library and document number in the LKR field (subfields \$l and \$b).

The parameter, PAR ,is set in column 3 of the `check_doc` table.

The check routine is set in `./<bib_library>/tab/check_doc`

!	1	2	3
!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!- !!!!!!!!!!!!!!!!!!!!!!!!!!!!>			
CATALOG-INSERT	check_doc_lkr		PAR

After the PAR is set in the `check_doc_lkr` routine, the system activates the `check_doc_lkr` routine and an additional routine to check if LKR \$\$aPAR has a corresponding LKR field in the linked document. This includes a validity check of LKR\$\$r. The check algorithm is as follows:

For each LKR \$\$aPAR with \$\$b<linked document number> and \$\$l<BIB Library> fields on document A:

- retrieve the linked document number (from \$\$b)
- retrieve the 7xx tag (from LKR\$\$r)
- check the linked document (document B) for a LKR field with the following information:
 - \$\$b with document A's system number
 - \$\$r with paired 7xx tags

The following are the possible 7xx tag pairs:

- 78000-78500
- 78005-78504
- 78006-78505
- 78004-78507
- 78007-78501
- 78001-78506
- 7750-7750
- 7870-7870
- 7670-7650
- 7760-7760

For example, if document A's LKR contains 78000 in \$\$r, then document B's LKR contains 78500 in \$\$r.

If there is no PAR in the linked record or there is no match in LKR\$\$r of the linked record (as defined in list of 7xx tag pairs), a check doc message 0217 is displayed. For example: “LKR – There is no PAR and/or invalid paired subfield "r" in the linked document USM01: 000000847”. This message is set in ./aleph/error_lng/check_doc.

```
!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
0217 L LKR - There is no PAR and/or invalid paired subfield "r" in
linked document $1:$2.
```

check_doc_locate

This program checks if there are records in the database that are similar to the record currently being updated. The mechanism used by this program is determined by the definitions in the `tab_locate` table of the library's tab directory.

check_doc_match

This program checks if there are records in the database that are duplicates of the one currently being edited. The mechanism used by this program is determined by the definitions in the `tab_match` table of the library's tab directory. Note that in the `tab_match` table, the match code 'CAT' (column 1 of the table) is used to specify the matching routines performed by the `check_doc_match` program.

check_doc_new_acc

This program checks whether or not a new record is opened in the headings list of the library.

check_doc_new_acc_aut

This program checks whether or not the cataloged heading is a new entry in the headings list of the bibliographic library or of the authority library.

check_doc_paired_fields This program checks the following two aspects related to fields that are linked by subfield \$6 (subfield \$6 contains data that links fields that are different script representations of each other), such as:

```
1001 L $$601$a[Name in Chinese script].
1001 L $$601$aShen, Wei-pin.
```

If the record contains a field with subfield \$\$6 <value numeric - other than '00'>, there must be an additional field with the same tag and indicators and the same \$\$6 value.

If the record contains a field with subfield \$\$6 <value numeric - other than '00'>, there must be no more than two fields that have the same \$\$6 value.

check_doc_tag_text

This program checks the validity of text entered into a subfield defined in the `check_doc_tag_text` table.

check_doc_unique_index

This program checks whether or not a duplicate record is opened in the Direct (Z11) index.

check_doc_url

This program checks the validity of an external URI/URL link from subfield \$u of fields such as 856, 505, 530, and so on. The default timeout for this program is 10

seconds. In order to change it, use the `setenv check_url_timeout` parameter in `pc_server_defaults`.

Note that adding the prefix `http://` is not mandatory. If no prefix is entered, the program will assume that the `http` protocol is used.

28.3 Check Programs For Document Deletion

Following are the available check programs for document deletion:

check_doc_delete_lkr

This program checks if there are any links from the record to be deleted to another record. It checks ADM, ITM, and HOL links from the record to records in other libraries.

check_doc_delete_lkr_no_ana

This program checks if there are any links from the record to another record in the same library that need to be deleted. It checks UP (up link), ANA (analytic) and PAR (parallel) links.

check_doc_delete_lkr_itm

This program checks if there are any ITM links from the record to be deleted to another record. In a MAB environment, it also checks links via the 090i field.

check_doc_delete_item

This program checks if the record to be deleted has any associated items.

check_doc_delete_item_opac

This program checks if the record to be deleted has any associated items that are viewable in the OPAC, i.e. column 10 of the `tab15.lng` line is set to Y.

check_doc_delete_order

This program checks if the record to be deleted has any associated order.

check_doc_delete_copies

This program checks if the record to be deleted has any associated subscriptions.

check_doc_delete_loan

This program checks if the record to be deleted has any associated items on loan.

check_doc_delete_hold

This program checks if the record to be deleted has any photocopy requests or/and hold requests associated to the items linked to it.

check_doc_delete_aut_bib

This program checks if the authority record to be deleted has any bibliographic records associated with the heading of the record.

check_doc_delete_object

This program checks if there are digital objects that are associated with the record being deleted. Note that an object is related to an ADM record depending on the value in the object's 'Cat. Sublibrary' field.

Note that in the Messages tab (lower pane) window of the Cataloging module, the View Related button is enabled with the following check routines:

- `check_doc_unique_index`

- check_doc_delete_lkr
- check_doc_locate
- check_aut_duplicate

The button retrieves the related record associated with the message displayed for the record being checked.

29 Fixed-length Fields Checking Routines

Fixed-length fields checking routines are table-driven. These routines are flexible and can be customized by the system librarian. Each fixed-length field has its own table for defining validation routines; the structure of the table is the same for all fields. Currently, fixed-length validation routine tables have been defined for MARC 21 006, 007, 008 and LDR (leader), and for UNIMARC 100 and LDR (leader).

The fixed-length tables for MARC 21 are the following:

- check_doc_field_006
- check_doc_field_007
- check_doc_field_008
- check_doc_field_ldr

The fixed-length tables for UNIMARC are the following:

- check_doc_field_100
- check_doc_field_ldr

Note that for the fixed-length validation checks to be functional, the check_doc_line program must be listed in the check_doc table located in the library's tab directory.

Following is a sample of a check_doc_field_<tag> table, (check_doc_field_006):

```

1   2   3   4   5   6   7
!!-!!!-!-!!!-!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
XX      000      1  acdefgijklmoprst

XX 000 a 001      1 ^abcdefghijklmnop|
XX 000 a 002      1 ^abcdefghijklmnop|
XX 000 a 003      1 ^abcdefghijklmnop|
XX 000 a 004      1 ^abcdefghijklmnop|
XX 000 a 001-004  3 check_val_left_just
3 check_val_alpha_order
XX 000 a 005      1 ^abcdefgj|
2 uv
XX 000 a 006      1 ^abcdfrs|
2 ghiz
XX 000 a 007      1 ^abcdefghijklmnopqrstvwz|
2 34hxy
XX 000 e 005-006  1 ^^,aa,ab,ac,ad,ae,af,ag,am,an,ap,au,az
1 ba,bb,bc,bd,be,bf,bg,bh,bi,bj,bo,br,bs
1 bu,bz,ca,cb,cc,ce,cp,cu,cz,da,db,dc,dd
1 de,df,dg,dh,dl,zz,||

```

Key to the check_doc_field_ tables:

Note that for columns that contain positions of the field a zero is added to the left of the position.

Column 1 - Record Format

Enter a specific record format, or use XX as a wildcard to indicate that the values for the position(s) of the field are appropriate for any format. Refer to for more information on record formats.

Column 2 - Match Offset

If needed, enter the field position used as a matching point for the character specified in column 3 (Start position). For example, in the section above:

```
XX 000 a 001      1 ^abcdefghijklmop|
```

^abcdefghijklmop| are valid values for position 01 of the 006 MARC 21 field when position 00 of the field contains value "a".

Column 3 - Match Character

If needed, enter the field character (value) of the position given in the previous column. For example, in the section above:

```
XX 000 a 001      1 ^abcdefghijklmop|
```

^abcdefghijklmop| are valid values for position 01 of the 006 MARC 21 field when position 00 of the field contains value "a".

Column 4 - (Start) Position

Start of the position range to check. For example, in the section above:

```
XX      000      1 acdefgijkmoprst
```

acdefgijkmoprst are the valid values for position 00 of the 006 MARC 21 field.

Column 5 - End Position

If needed, end of the position range to check. For example, in the section above:

```
XX 000 e 005-006 1 ^^,aa,ab,ac,ad,ae,af,ag,am,an,ap,au,az  
1 ba,bb,bc,bd,be,bf,bg,bh,bi,bj,bo,br  
1 bs,bu,bz,ca,cb,cc,ce,cp,cu,cz,da,db  
1 dc,dd,de,df,dg,dh,dl,zz,||
```

^^,aa,ab,ac,ad,ae,af,ag etc. are the valid values for positions 05 to 06 of the 006 MARC 21 field when position 00 of the field contains value "a".

Column 6 - Check Type

Defines the type of check that should be applied. Values are 1, 2 and 3:

1 = Check for valid values.

2 = Check for obsolete values.

3 = Run an external check program.

Column 7 - Check Values

The check values depend on the check type defined in column 6.

If the check type is 1, then this column contains the list of valid values that are valid for the position range. If the value being checked is present on the list, then no error message is displayed.

Values are separated by commas. If the position range is only one character wide, the commas can be omitted. For example, in the section above:

```
XX      000      1  acdefgijklmoprst
```

acdefgijklmoprst are the valid values for position 00 of the 006 MARC 21 field.

If the check type is 2, then this column contains a list of values that can be present but are obsolete. If the value being checked is present on the list, an error message will be displayed informing the cataloger that the value is obsolete.

Values are separated by commas. If the position range is only one character wide, the commas can be omitted. For example, in the section above:

```
XX 000 a 006      1 ^abcdfrs|
                   2 ghiz
```

^abcdfrs| are valid values for position 06 of the 006 MARC 21 field when position 00 of the field contains value "a". ghiz are obsolete values for the same position.

Note that for values of type 1 and 2, the blank should be indicated using the DOC-BLANK-CHARACTER variable in the tab100 table of the library's tab directory.

If the check type is 3, then this column contains the name of the external check routine that must be performed for the position range. For example, in the section above:

```
XX 000 a 001-004 3 check_val_left_just
```

In this instance, for positions 01 to 04 of the 006 MARC 21 field, when the value of position 00 is "a", the system performs the check_val_left_just checking routine. This program verifies that the values in the position range are left-justified.

Following are ALEPH's external check routines for fixed-length field validation tables:

check_val_left_just

Verifies that the values in the position range are left-aligned.

check_val_alpha_order

Verifies that the values in the position range are in alphabetical order, ignoring spaces.

check_val_run_time

Verifies that the three characters specified constitute a valid running time (that is, 000-999, ---, and nnn).

check_fixed_field_length

Verifies that the field is as long as the start and stop offsets would indicate.

check_val_red_ratio

Verifies positions 06-08 of the 007 MARC 21 field for microforms (reduction ratio).

The specific reduction ratio of the microform, recorded as three digits. The number is right-justified and each unused position contains a zero. A hyphen is used for any unknown portion of the reduction ratio.

check_val_date_6

Verifies positions 17-22 of the 007 MARC 21 field for motion pictures (film

inspection date). Six characters that indicate the most recent film inspection date; the date is recorded in the pattern ccyymm (century/year/month). A hyphen is used for any unknown portion of the date. Six fill characters (|||||) are used if no attempt is made to code these character positions.

check_val_blank

Verifies that the position range consists only of blanks (^).

check_val_date_4

For a four-position range, verifies that it forms a valid date (i.e., 1999, 19uu, ||||, and so on. The program does not permit the first position to be "u", nor does it allow "uuuu". This routine is only relevant for the bibliographic (for example, USM01) library - the 008 fixed-length field, "Date 1" and "Date 2" subfields, positions 07-10 and 11-14.

check_val_all_9

Verifies that the position range consists only of the digit 9.

check_val_date_8

For an eight-position range, verifies that it forms a valid date (yyyymmdd). The program allows the last two positions (corresponding to the day) to be blanks.

check_val_numeric

Verifies that the position range consists only of digits. This routine is only relevant for the holdings (for example, USM60) library - the 008 fixed-length field, "Number of copies reported" subfield, positions 17-19 and "Date of report" subfield, positions 26-31

check_val_all_u

Verifies that the position range consists only of the value "u".

check_val_date_4_or_u

Like check_val_date_4, for a four-position range, verifies that it forms a valid date. The difference is that "uuuu" is permitted.

check_val_country

Verifies that the position range forms a valid MARC country code. Valid country codes are defined in the `marc_country_codes` table in the `alephe/tab` directory. This routine is relevant for a bibliographic library (for example, USM01) - the 008 fixed length field, "Publication" field, positions 15-17

check_val_language

Verifies that the position range forms a valid MARC language code. Valid language codes are defined in the `marc_language_codes` table in the `alephe/tab` directory. This routine relates to the following:

Bibliographic library (for example, USM01) - the 008 fixed length field, "Language" field, positions 35-37

Holdings library (for example, USM60) - the 008 fixed length field, "Language" field, positions 22-24.

check_val_bitdepth

Verifies positions 06-08 of the 007 MARC 21 field for ELECTRONIC RESOURCE (image bit depth).

A three-character number specifying the exact bit depth of the scanned image(s) that comprise(s) the computer file, or a three-character alphabetic code which indicates that the exact bit depth cannot be recorded. Since the exact bit depth is useful, coding should not include missing digits represented by hyphens (-). Three fill characters (|||) are used when no attempt has been made to encode this data element.

check_val_heading_use

Verifies that there is at least one "a" in the headings use codes of MARC 21 008 field (positions 14-16). This routine checks if the heading has been marked valid for any use. This routine should only be used for Authority (for example, USM10) libraries.

30 Validation Messages (Table-dependent)

The `check_doc.lng` table of the library's tab directory contains user-defined validation messages that are table-dependent. For example, this table is used to define the error messages that are displayed when performing the `check_doc_doc` checks.

The following is a sample line from the `check_doc_doc` table:

```
OC XX 5002 01 01 245##
```

The message 5002 for the above example must be defined in the `check_doc.lng` table. Following is a sample of the table that contains the message for the line from the `check_doc_doc` table:

```
! 1 2 3
!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

5001 L Multiple 1XX! A record cannot have more than 1 main entry.
5002 L Required 245 field is either missing or duplicated.
5003 L Required 260 field is either missing or duplicated.
5007 L Required 310 field is either missing or duplicated.
```

Key to the `check_doc.lng` table:

Column 1 - Error Message Number

Error message number. Must be between the range of 5000-7000.

Column 2 - ALPHA

ALPHA code. Must always be L.

Column 3 - Error Message Text

Enter the message text that is displayed in the Cataloging module when performing the check routine.

31 Validation Messages (System-driven)

The `check_doc` table in the `$aleph_root/error_lng` directory provides for validation messages for the check doc programs. The error messages defined in this table are system-driven and are between the range of 0001-4999 and of 9000-9999.

Following is a section of the `check_doc` table:

```
1 2 3
```

!!!!-!-!!>

0001 L Document number \$1 in library \$2 points to current document.
0002 L Document has \$3 item(s) attached to ADM record \$1 in library
\$2
0003 L ADM record has \$1 item(s) attached.
0004 L Document has \$3 order(s) attached to ADM record \$1 in library
\$2
0005 L ADM record has \$1 order(s) attached.

Key to the check_doc table:

Column 1 - Error Message Number

Error message assigned by ALEPH.

Column 2 - ALPHA

ALPHA code. Must always be L.

Column 3 - Error Message Text

Message text that is displayed in the Cataloging module when performing the check routine.

32 Cataloging Productivity Report

A cataloging productivity report can be produced by running the Count of New and Updated Catalog Records - by Cataloger (com-02) batch service from the Services menu.

This service measures the productivity of the catalogers within a specific time period. The report includes the number of new records cataloged and the number of updated records by each cataloger. In addition, this service summarizes the total cataloging activity (total number of new records and total number of updated records) for the library between the given time period. Deleted records are also included in the updated records counts.

Alternatively, the Report of New and Updated Catalog Records by Cataloger (com-03) service can be used as well. This service works in a similar way to the com-02 service. Its enhanced algorithm is based on the oracle table, CAT Fields (Z106).

32.1 HOL Records tab of Records Editor

The HOL Records tab, found in the lower pane of the Records Editor, can be set up to include additional information about the HOL record, based on the 801 paragraph of the edit_paragraph.lng table that is in the HOL library's data_tab directory. For example, set the edit_paragraph table with:

```
801 OWN## D      :^  
801 852## 3
```

And the pc_tab_col.lng table with

```
PC_COM_HOL_SELECT L HOL Information 04 050 01 C01 HOL  
edit_doc_paragraph 801
```

33 Column Headings (pc_tab_col.lng and tab_col.dat)

The pc_tab_col.lng table of the library's tab directory and the tab_col.dat table of the ALEPHCOM/LNG/TAB directory define the columns of information that are displayed in list windows in the GUI clients.

In order to define column headings, edit the bibliographic library (for example, USM01) table pc_tab_col.lng using the ALEPHADM module. Note that some list window columns are not controlled by this table. They are controlled by the tab_col.dat table on the GUI client.

For more information about pc_tab_col.lng, see the ALEPH User Guide - General chapter - Desktop Customization - GUI and Toolbars section.

The following is a list of the Cataloging windows which use the pc_tab_col.lng table for formatting data and their identifiers (Column 1 in pc_tab_col.lng):

Identifier	Cataloging GUI Windows
PC_CAT_SCAN	Headings in Library (Search headings options)

* In this GUI table, an optional color/font can be used by the system for color/font differentiation between values of the same column. The alternative font and color are defined in Columns 8 and 9 of pc_tab_col.lng.

The following is a list of the Cataloging windows that use the tab_col.dat table for formatting data and their identifiers:

Identifier	Cataloging GUI Windows
CAT_SCAN_LIB_LIST	Choose Library (Search headings options)

34 Default Values for Fixed Fields in New Records

The default values for the 008 and the LDR MARC 21 fixed-fields and for the 100 and the LDR UNIMARC fixed-fields are hard-coded. You can manipulate these values by using the tab_tag_text table of the library's tab directory. This table is used to define for each field, according to the record format, the default value for these fields when records are created in the system.

Following is a sample of the tab_tag_text table:

```

!1  2 3
!!!-!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!->
LDR BK ^^^^^nam^a22^^^^^^a^4500
LDR SE ^^^^^nam^a22^^^^^^a^4500

008 BK ^^^^^s2000^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^000^^eng^d

008 SE ^^^^^c19009999^^^^r1^^^^^^^^0^^^^0eng^d

```

Key to the tab_tag_text table:

Column 1 - Field Code

Enter the field code of the field for which the default values are being defined (for example, LDR, 008 or 100).

Column 2 - Record Format

Enter a specific record format to indicate the format for which the default values for the field are appropriate.

Column 3 - Default Values

Enter the default values for the field. Use the blank character specified in the DOC-BLANK-CHAR variable of the tab100 table to represent spaces.

35 Importing Records

The conversion mechanism of the Import records option of the Cataloging module is based on two different types of conversions:

- Local conversions (that is, conversions performed at the level of the client)
- Remote conversions (that is, conversions performed by server routines)

For both types of conversions, you must define the conversion program under the `convert.ini` file in the following format:

```
!   1           2           3           4
!-----!-----!-----!-----
---->
LOCAL      BIP           0           Books in print conversion
LOCAL      CDMARC        0           CDMARC conversion
LOCAL      2709POR       5           2709 Portugal conversion
LOCAL      2709OCLC      3           2709 OCLC conversion
LOCAL      CDMARC        CP1251     CDMARC conversion for Cyrillic
LOCAL      2709OCLC      9           2709 OCLC conversion with atl fix
LOCAL      RLINPASS      5           RLIN Pass
LOCAL      2709OCLC      CP936      2709 OCLC USMARC conversion for
Chinese(simp)
LOCAL      2709OCLCUNIMARC CP936      2709 OCLC UNIMARC conversion for
Chinese(simp)
LOCAL      2709OCLC      CPUTF      2709 OCLC USMARC conversion for
UTF
LOCAL      2709OCLCUNIMARC CPUTF      2709 OCLC UNIMARC conversion for
UTF
REMOTE     SEQ           ALEPH Sequential
REMOTE     SEQ300        ALEPH300 Sequential
REMOTE     MARC           MARC
```

Column 1 - Conversion Program

Local or Remote

Column 2 - Conversion Type

Among the possible types for local program are: 2709POR, 2709OCLC, 2709OCLCUNIMARC, BIP, CDMARC, RLINPASS

Column 3 - Character Conversion Type

Code Page - Not used in REMOTE program. This is the identifier of the conversion that is being performed

Column 4 - Conversion Name

Conversion Name

35.1 Remote Conversions

For remote conversions, the following are the conversion programs that are currently available:

pc_cat_conv_mab_d: conversion from MAB2 Diskettenformat (ekz).

pc_cat_conv_cdmarc: conversion from CDMARC.

pc_cat_conv_aleph_seq: conversion from ALEPH Sequential format.

pc_cat_conv_aleph300_seq: conversion from ALEPH 300 Sequential format. Note that only single records can be loaded/converted by this program.

pc_cat_conv_marc: conversion of MARC records separated either by a new line character, or by a MARC record separator (ASCII - 29 or Hexadecimal - 1D). It can also be used for the conversion of a single MARC record. We recommend that you use this program instead of the '2709 OCLC conversion' local conversion program as that can be problematic in the case of long fields (longer than 2000 characters).

The conversion specifications are defined in the `pc_tab_cat_conv` table, located in the library's `tab` directory. The following is a sample of the table:

1	2	3
!!!!!!!-!!!!!!!>		
MAB	pc_cat_conv_mab_d	850_TO_UTF
MARC	pc_cat_conv_marc	
CDMARC	pc_cat_conv_cdmarc	
SEQ	pc_cat_conv_aleph_seq	
SEQ_8859_1	pc_cat_conv_aleph_seq	8859_1_TO_UTF
SEQ300	pc_cat_conv_aleph300_seq	ALEPH300_TO_UTF

Key to the `pc_tab_cat_conv` Table:

Column 1 - Conversion Routine

This is the identifier of the conversion that is being performed (free-text).

Column 2 - Conversion Program

Enter the conversion program that should be performed for the specific conversion routine defined in column 1.

Column 3 - Parameters Certain conversion routines require additional information, such as character conversion routines. This column is used to define additional parameters for conversion programs.

36 Combining Diacritics

The following description is relevant if the combined character functionality is not supported (in `alephcom.ini` `CombinedCharSupported=N`). If the combined character functionality is supported (`CombinedCharSupported=Y`), you can enable the display of hidden characters by selecting the Show hidden characters option in the Edit text menu (`Alt+F2`).

In order that combining diacritical marks be displayed as clearly as possible in the cataloging draft, they can be displayed in conjunction with a "spacing" character (similar to the way that they are displayed in the Combining Diacritical Marks table in "The Unicode Standard"). The underline (`U+005F`) has been chosen for diacritics that

are positioned above the character, and the dotted circle (U+25CC) has been chosen for combining diacritics that are positioned below the character.

```
·  
ė ë ẻ e̊
```

The spacing characters and the diacritics with which they are used are defined in the `Spacer.ini` table located in the `Alephcom/tab` directory. Following is an extract of this file:

```
! 1      2      3  
!!!!-!!!!-!!!!  
005F 0300 0315  
005F 0334 0338  
25CC 0316 0333  
25CC 0339 033C
```

In the extract above, the underline (U+005F) has been selected as the spacer for the following ranges of Unicode characters:

0300 to 0315

0334 to 0338

In addition, the dotted circle (U+25CC) has been selected as the spacer for the following ranges of Unicode characters:

0316 to 0333

0339 to 033C

37 Record Length Limits

Records in ALEPH are limited to:

5000 subfields.

45000 bytes.

Each field is limited to 2000 bytes.

38 Hidden Fields

Hidden fields are fields that are present in the cataloging record but that are not displayed in the Catalog Editor. In other words, these are fields that cannot be updated directly by the cataloger through the Cataloging module. To define a field as hidden, add the field to the `tab_cat_hidden_fields` table located in the library's `tab` directory.

Following is a sample of the `tab_cat_hidden_fields` table:

```
! 1  
!!!!
```

39 Record Manager

The Record Manager displays information regarding the record currently being edited in the Catalog Editor (upper pane of the Cataloging tab). The information is displayed in tree structure after opening a record from the server or after saving a new record. Note that the display in the Record Manager is limited to 800 lines.

The system librarian is in charge of defining the following:

The display of the relation between the holdings and the items records. This is determined by the *item_hol_tree_style* variable of the *pc_server_defaults* file. The following options are available:

- 1 - Do not display the connection between the items and the linked holdings records
- 2 - Display the items under the related administrative (ADM) record and also under the linked holdings record
- 3 - Display the items only under the linked holdings record (in this case the holdings libraries nodes are displayed before administrative - ADM - libraries nodes)

The number of leaves in each node is determined by the *pc_tree_view_max_branch* variable of the *pc_server_defaults* file. In the following example, nodes stemming from BIB records (in the Record Manager) are limited to 10 leaves and nodes stemming from the Administration/Holdings environment (under the Items, Acquisitions and Circulation_overview trees) are limited to three leaves::

```
setenv pc_tree_view_max_branch 10
```

This variable is limited to 750 nodes.

The way in which the tree is displayed when first opened. This is determined by the *expand_tree_style* variable of the *pc_server_defaults* file. The following options are available:

- 1 - Expand only the selected record node.
- 2 - Expand only record nodes (administrative, bibliographic and holdings)
- 3 - Expand all existing nodes (administrative, bibliographic, holdings, item, order, and so on.)

Note

The display of the holdings record is also based on the *OWN field*, and the display of the items on the ITEM-SHOW permission in the *user_function.lng* table.

40 Overview Tree

The Overview Tree shows the records in the system that are related to a cataloging record. For example, it displays the holdings records, the administrative record, and the items, subscriptions, orders and loans attached to the selected cataloging record. The information is displayed in tree structure.

Note

The display in the Overview Tree is limited to 800 lines.

The system librarian is in charge of defining the following:

The display of the relation between the holdings and the items records. This is determined by the *item_hol_tree_style* variable of the *pc_server_defaults* file. The following options are available:

- 1 - Do not display the connection between the items and the linked holdings records
- 2 - Display the items under the related administrative (ADM) record and also under the linked holdings record
- 3 - Display the items only under the linked holdings record (in this case the holdings libraries nodes are displayed before administrative - ADM - libraries nodes)

Note
The display of the holdings record is also based on the OWN field, and the display of the items on the ITEM-SHOW permission in the *user_function.lng* table.

The number of leaves in each node. This is determined by the *pc_tree_view_max_branch* and *pc_filter_tree_view_max_branch* variables of the *pc_server_defaults* file. In the following sample, nodes of Bibliographic records are limited to 10 leaves and nodes stemming from Administration/Holdings records are limited to three leaves:

```
setenv pc_tree_view_max_branch      10
setenv pc_filter_tree_view_max_branch 3
```

Both variables are limited to 750 nodes.

The way in which the tree is displayed when first opened. This is determined by the *expand_tree_style* variable of the *pc_server_defaults* file. The following options are available:

- 1 - Expand only the selected record node.
- 2 - Expand only record nodes (administrative, bibliographic and holdings)
- 3 - Expand all existing nodes (administrative, bibliographic, holdings, item, order, and so on.)

You can expand ALL nodes of a record in the Overview Tree, by clicking Expand All Nodes from the right-click menu. This action is the same as the initial state of a record tree when the environment variable *expand_tree_style* is set to 3.

The moving routines that are performed when records are moved through the Overview Tree. These routines are defined in the *tab_move_record* table of the bibliographic library's tab directory.

The sort of the items in the tree is dependent on the TREE function in the *tab_z30_sort* table of the administrative library.

The following is an excerpt of the *tab_move_record* table:

```
! 1          2          3
!!!!!!!!!!!!-!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ADM          ADM          move_adm_to_adm
```


Z30	ADM	move_z30_to_adm
Z16	ADM	move_z16_to_adm
Z68	ADM	move_z68_to_adm
ITEMS	ADM	move_items_to_adm
COPIES	ADM	move_copies_to_adm
ORDERS	ADM	move_orders_to_adm
HOL	BIB	move_hol_items_to_bib
Z30	HOL	move_z30_to_hol

Key to the tab_move_record Table:

Column 1 - Moving From Record Type of record to be moved. The available options are: ADM (administrative), Z30 (a single item), Z16 (a single subscription), Z68 (a single order), HOL (a single holdings record), ITEMS (all items under the Items node), COPIES (all subscriptions under the Subscriptions node), ORDERS (all orders under the Orders node), HOL-ITEMS, MEX.

Column 2 - Moving To Record Record to which the selected record is being moved. Available options are: ADM (administrative), BIB (bibliographic), HOL (holdings), HOL-ITEMS.

Column 3 - Moving Procedure Moving routines. The following are the available moving programs:

- **move_adm_to_adm**: moves all instances under an administrative record to another administrative record. This move includes:

- Items and item history (Z30 and Z30H)
- Holding requests and holdings request history (Z37)
- Loans and loan history (Z36)
- Photocopy requests (Z38)
- Advance booking - time slots (Z320)
- Short loans - status (Z321)
- Serials claim (Z20)
- Maintenance Records (Z328)
- Linked HOL records
- Subscription information (Z16)
- Routing lists (Z18)
- Members of routing lists (Z14)
- Routing trace (Z22)
- Orders (Z68)
- Order and subscription log (Z71)
- Arrival information (Z78)
- Acquisition claims (Z501)
- Budget transactions (Z601)
- Invoice - Line item (Z75)

The move is not performed if one of the items is linked to an incoming or to an outgoing ILL request.

- **move_adm_to_bib**: moves an ADM record from one BIB record to another BIB record. This move includes linked HOL records. It is not performed if an ADM record linked to the target BIB already exists (for example, a new BIB without an ADM).

- **move_bib_to_bib**: moves BIB and AUT records. Linked ADM and/or HOL records will be moved according to move_adm_to_bib and move_hol_to_bib. This procedure works only in MAB. When a holdings record is moved from one bibliographic record to another, all its attached items are moved as well. This program also moves the digital objects attached to the bibliographic record.

- **move_hol_to_bib**: moves a HOL record from one BIB record to another BIB record. It is not performed if an item is linked to the HOL. In this case, the Z30/ADM must be moved first.

This program works only in a multi-HOL, multi-ADM environment and requires the following setup:

ADM XXX60 XXX50

move_z30_to_adm: moves a selected item to another administrative record. This move includes:

Items and item history (Z30 and Z30H)
Holding requests and holding requests history (Z37)
Loans and loan history (Z36)
Photocopy requests (Z38)
Advance booking - time slots (Z320)
Short loans - status (Z321)
Serials claim (Z20)
Maintenance Records (Z328)

The move is not performed if:

- The item is linked to an incoming or to an outgoing ILL request.
- The item is linked to a holdings record.
- The item is linked to a subscription record.
- The item is linked to an order record.

- **move_z16_to_adm**: moves a selected subscription to another administrative record. This move includes:

Subscription information (Z16)
Items and item history (Z30 and Z30H)
Routing lists (Z18)
Members of routing lists (Z14)
Routing trace (Z22)

The move is not performed if:

- The item is linked to an incoming or to an outgoing ILL request.
- The item is linked to a holdings record
- The item is linked to an order record

- **move_z68_to_adm**: moves a selected order to another ADM record. This move includes:

Orders (Z68)
Items and item history (Z30 and Z30H)
Order and subscription log (Z71)
Arrival information (Z78)
Acquisition claims (Z501)
Budget transactions (Z601)
Invoice - Line item (Z75)

The move is not performed if:

- The item is linked to an incoming or to an outgoing ILL request.
- The item is linked to holdings record
- The item is linked to a subscription record

- **move_items_to_adm**: moves all items under the selected items node to another administrative record. The move includes all instances specified under `move_z30_to_adm`.

- **move_copies_to_adm**: moves all subscriptions under the selected subscriptions node to another administrative record. The move includes all instances specified under `move_z16_to_adm`.

- **move_orders_to_adm**: moves all orders under the selected orders node to another administrative record. The move includes all instances specified under `move_z68_to_adm`.

- **move_hol_items_to_bib**: moves a HOL record (together with its items, if any) to another BIB record. Moving of items is performed according to the same guidelines as in `move_z30_to_adm`; naturally, in this case the restriction preventing the move of an item connected to a HOL record is irrelevant.

- **move_z30_to_hol**: moves an item connected to a HOL record, to another HOL record connected to a different BIB record. Moving of the item is performed according to the same guidelines as in `move_z30_to_adm`; as in `move_hol_items_to_bib`, the HOL link restriction is not activated.

Note

If the relevant program for a particular move is not listed in the table, an error message is displayed in the GUI when trying to perform the selected move and the move will not be performed.

The following privileges are related to the functionality of the Overview Tree:

- Global permission to move records: CATALOG (MOVE-TREE-ITEM)
- Permission to move subscription records: CATALOG (MOVE-Z16)
- Permission to move order records: CATALOG (MOVE-Z68)
- Permission to move item records: CATALOG (MOVE-Z30)

No special permission is needed in order to view the Overview Tree.

41 Setting Up a Script for the Correction of Records in Aleph Sequential Format

41.1 Generic Fix Doc Script Specification

The Modify MARC Record File service (p_file_08) modifies records in ALEPH sequential format according to a user-specified processing script. This document describes the format of this script.

The script is in ALEPH table format, with 9 columns. The library can use multiple scripts by creating multiple tables, with different names. One of the parameters of the p_file_08 batch job is the name of the script (table). Blank lines and lines beginning with ‘!’ are ignored.

Col. 1	Iteration	1 digit
Col. 2	Tag	5 characters, tag and indicators. ‘#’ in any position acts as a wildcard)
Col. 3	Format Filter	2 characters, ‘#’ is a wildcard
Col. 4	First Position Filter	1 character
Col. 5	Position Range Start	blank or 3 digits
Col. 6	Position Range End	blank or 3 digits
Col. 7	Occurrence Filter	blank, a 5-digit number, or “FIRST”, “LAST”, “NOT-F”, or “NOT-L”)
Col. 8	Operation code	30 characters; the list of valid operation codes is in the <i>Generic Fix Doc Operations</i> section
Col. 9	Operation parameters	100 characters max; the list of valid parameters for each operation is listed below

41.2 Script Flow

The script processes every record in the input to p_file_08 as follows:

1. The operations are done in order of Iteration, then in the order within every Iteration. In other words, operations in Iteration 1 are processed before those in Iteration 2. Within Iteration 1, the operations are processed in the order listed in the script.
2. With the exception of the SORT-FILES operation, each operation is applied only to the fields whose tag matches the pattern in Tag.
 - a. If the Format Filter is not blank, the operation is done only if the current format of the record matches (as stored in the FMT field).
 - b. If the Occurrence Filter is not blank, only the specified occurrence or occurrences of the field are processed. Note that “NOT-F” means “not first” and “NOT-L” means not last.

- c. If the First Position Filter is not blank, the operation is performed only if the first position in the field matches the pattern in Tag.
- 3. If an operation deletes a field, then that field does not exist for subsequent operations.
- 4. Certain operations on fixed fields act only on the position range specified by Position Range Start and Position Range End. Note that field positions are counted starting from zero, following the MARC 21 convention but contrary to the UKMARC convention.

41.3 Generic Fix Doc Operations

ADD-FIELD

Adds a field to the record with the coding and contents specified. An occurrence of the new field is added to the record for each existing occurrence of the tag specified in col.2. To add a field once to all records in a file, use a mandatory non-repeatable tag (for example, LDR) in col.2.

Parameters: (comma-separated)

1. field code (5 characters: tag plus indicators)
2. field alpha (1 character)
3. field contents (include subfield codes, for example, \$\$a)

ADD-FIELD-GENERAL

Similar to the ADD-FIELD operation with the ability to specify the delimiter (rather than using comma). It is not recommended to use a space as a delimiter. The first character in the operation parameter indicates the delimiter.

The following example uses the ^ sign as a delimiter:

```

1 2 3 4 5 6 7 8 9
1 LDR ADD-FIELD-GENERAL ^655 4^L^$$aGuides, vade-mecum, etc.

```

ADD-CURRENT-DATE

Add a new field to the document that includes the current date in the YYYYMMDD format.

To activate the new functionality, add a line such as the following to the configuration file under the directory ./<BIB library>/tab/import:

Parameters: (comma-separated)

1. field code (5 characters: tag plus indicators)
2. field alpha (1 character)
3. field contents (include subfield codes, for example, \$\$a)

ADD-SUBFIELD

Adds a subfield to the specified field. An occurrence of the subfield is added to all occurrences of the tag specified in column 2.

Parameters: (comma-separated)

1. subfield code
2. contents

ADD-SUBFIELD-GENERAL

Similar to the ADD-SUBFIELD operation with the ability to specify the delimiter (rather than using comma).

The first character in the operation parameter indicates the delimiter. It is not recommended to use a space as a delimiter.

The following example uses a ^ sign as a delimiter.

```

1 245## ADD-SUBFIELD-GENERAL ^c^prepared by the chief of the Bureau of Stat
cs, Treasury Department.

```

CHANGE-FIELD

Changes the tag of a field.

Parameters: 1. field tag (3 characters)

CHANGE-FIRST-IND

Changes the value of the first indicator of a variable field.

Parameters: (comma- or space-separated)

1. value to match; '#' acts as a wildcard (1 character)
2. value to set indicator to upon a match (1 character)

CHANGE-FIRST-IND-MATCH

Changes the value of the first indicator of a variable field. It is similar to CHANGE-FIRST-IND except that it has a third parameter, which is a string to match upon. If there is such a string in the tag's content, then the indicator is changed as specified.

Parameters: (comma- or space-separated)

1. value to match; '#' acts as a wildcard (1 character)
2. value to set indicator to upon a match (1 character)
3. string to match upon. It can include subfield delimiters (\$\$) and wildcards (#).

CHANGE-FIRST-IND-MATCH-GENERAL

Similar to the CHANGE-FIRST-IND-MATCH operation with the ability to specify the delimiter (rather than using comma or space).

The first character in the operation parameter indicates the delimiter. It is not recommended to use a space as a delimiter.

The following example uses the ^ sign as a delimiter.

```

1 100## CHANGE-FIRST-IND-MATCH-GENERAL ^#^1$$Altman, Philip L.

```

CHANGE-SECOND-IND

Changes the value of the second indicator of a variable field.

Parameters: (comma- or space-separated)

1. value to match; '#' acts as a wildcard (1 character)
2. value to set indicator to upon a match (1 character)

CHANGE-SECOND-IND-MATCH

Changes the value of the second indicator of a variable field. It is similar to CHANGE-FIRST-IND except that it has a third parameter, which is a string to match upon. If there is such a string in the tag's content, then the indicator is changed as specified.

Parameters: (comma- or space-separated)

1. value to match; '#' acts as a wildcard (1 character)
2. value to set indicator to upon a match (1 character)
3. string to match upon. It can include subfield delimiters (\$\$) and wildcards (#).

CHANGE-SECOND-IND-MATCH-GENERAL

Similar to the CHANGE-FIRST-IND-MATCH operation with the ability to specify the delimiter (rather than using comma or space).

The first character in the operation parameter indicates the delimiter. It is not recommended to use a space as a delimiter.

The following example uses a ^ sign as a delimiter.

```
1 650# CHANGE-SCND-IND-MATCH-GENERAL ^#^0^$$vHandbooks, manuals, etc
```

CHANGE-SUBFIELD

Changes every occurrence of a subfield code in a variable field to another value.

Parameters: (comma- or space-separated)

1. subfield code to match; there is no wildcard (1 character)
2. value to change subfield code to upon a match (1 character)

CONCATENATE-FIELDS

Concatenates the first occurrence of a tag all occurrences of another given tag.

Parameters (comma-separated):

1. Field code (5 character: tag plus indicators)
3. Subfields to concatenate (list of subfields is a single string without comma delimiters)

COND-LOAD-VAL-POS

Determines whether to continue processing the record or reject it, based on the value of the field position (col.5) in a fixed field.

Parameters: (comma-separated)

Condition type; can be "Y" or "N":

If "Y" and the value of the field position (col.5) is in the list supplied in the second parameter, the record is rejected.

If “N” and the value of the field position is not in the list, the record is also rejected. In all other cases, the record is accepted. (1 character)

List of values of the field position that determine whether or not to accept the record

COND-LOAD-VAL-FIELD

Determines whether to continue processing the record or reject it, based on the presence or absence of a particular field tag

Parameters:

Condition type; can be “Y” or “N”:

COND-LOAD-VAL-MATCH

Determines whether to continue processing the record or reject it, based on the presence or absence of a particular field tag + subfield + content string

Parameters: (comma-separated)

1. Condition type; can be “Y” or “N”:
2. Text string in the field that determines whether or not to accept the record

COPY-FIELD

Copies the entire contents of each matching field to another field. Any attempt to copy to the same tag, or to a tag that matches the pattern in col.2 is ignored in order to prevent an infinite loop. (If it is necessary to duplicate a field, tag and all, first COPY-FIELD to a temporary tag, then CHANGE-FIELD to the desired tag.)

Parameters: (comma-separated)

field code of new field (5 characters: tag plus indicators)

field alpha (1 character) (defaults to L)

COPY-SYSTEM-NUMBER

Copies the entire contents of a fixed length control field to new variable field and subfield, optionally adding a prefix.

Parameters: (comma-separated)

field code of new field (5 characters: tag plus indicators)

field alpha (1 character)

new subfield (1 character)

optional prefix to add to the contents of the fixed length control field after they are copied to the new variable field

DELETE-FIELD

Unconditionally deletes a fixed or variable field.

Parameters: *none*

DELETE-FIELD-COND

Deletes a variable field if it contains the specified string. Matching is exact and case-sensitive.

Parameters: (comma-separated)
condition type; can be “Y” or “N”:

If “Y” and the match string is present in the field, the field is deleted.

If “N” and the match string is not present in the field, the field is also deleted. In all other cases, the field is retained.

match string

DELETE-FIELD-COND-GENERAL

Similar to the DELETE-FIELD-COND operation with the ability to specify the delimiter (rather than using a comma).

The first character in the operation parameter indicates the delimiter. It is not recommended to use a space as a delimiter.

The following example uses the ^ sign as a delimiter.

```
1 60010          DELETE-FIELD-COND-GENERAL          ^Y^Austin, Mary Hunter, ■
```

DELETE-FIXED-COND

Deletes a fixed field if the specified position (col.5) or range (col.5-6) matches the pattern given.

Parameters: (comma-separated)
condition type; can be “Y” or “N”:

If “Y” and the match string pattern matches the values present in the position range, the field is deleted.

If “N” and the match string pattern does not match the values present in the position range, the field is also deleted. In all other cases, the field is retained.

match string; ‘#’ in any position is interpreted as a wildcard.

DELETE-SUBFIELD

Removes all occurrences of the specified subfield and contents from the variable field. If the last subfield is removed, the entire field is deleted.

Parameters: 1. subfield to remove (1 character)

DELETE-SUBFIELD-DELIMITER

Removes all occurrences of the specified subfield delimiter (for example, , \$\$a) only. The delimiter is replaced with a single space. The delimiter of the first subfield in the field will not be removed.

Parameters: 1. subfield delimiter to remove (1 character)

EDIT-SUBFIELD-HYPHEN

Inserts a hyphen if it is not already present at the specified position within each occurrence of the specified subfield. An insertion does not take place if the existing contents are not long enough.

Parameters: (comma-separated)
subfield in which to insert the hyphen (1 character)
position within subfield at which to insert the hyphen (3 digits, leading zeroes required)

FIXED-CHANGE-VAL

Changes the value of the specified range of positions (cols.5-6) in a fixed field if the current value matches a pattern.

Parameters: (comma-separated)
pattern; '#' is a wildcard. Note that the pattern must be exactly as long as the specified position range.
replacement values. Must be exactly as long as the specified position range.

FIXED-CHANGE-VAL-RANGE

Replaces every occurrence of a character found anywhere in the specified range of positions (cols.5-6) in a fixed field with another character.

Parameters: (comma-separated)
character to match; '#' is a wildcard
replacement character (use ^ for blank, | for fill character)

FIXED-FIELD-EXTEND

Extends a fixed field if it is at least the specified minimum length but less than the maximum, by appending the specified character to bring it up to the maximum length.

Parameters: (comma-separated)
minimum length of field required for it to be extended (3 digits, leading zeroes required)
length to extend field to (3 digits, leading zeroes required)
character to pad field with (1 character, use ^ for blank, | for fill character)

FIXED-RANGE-OP

Performs the specified operation on a position range (cols.5-6) of a fixed field.

Parameters: 1. field operation. Must be either LOWER or LJ.
LOWER: changes all characters in the range to lowercase
LJ: left-justifies the non-blank values in the range

REPLACE-STRING

Replaces all occurrences of the specified string with another within a variable field. The strings can include subfield codes.

Parameters: (comma-separated)
pattern to match; '#' acts as a positional wildcard; otherwise, matching is exact and case-sensitive.
replacement string; may be the empty string

REPLACE-STRING-GENERAL

Replaces all occurrences of the specified string with another, using a pre-defined delimiter between the source and target strings. The strings can include subfield codes.

The delimiter between the source and target string is set in the position 1 of the operation parameter. From position 2 and onward, set the source and target strings separated by the delimiter character set in position 1.

Parameters: first position in the operation parameter indicates the delimiter. From position 2 and onward, set the source and target strings separated by the delimiter character set in position 1.
 pattern to match; '#' acts as a positional wildcard; otherwise, matching is exact and case-sensitive.
 replacement string; may be the empty string

SORT-FIELDS

Sorts the fields in the record by tag number, based on the usual sort order for the ALEPH library in which this script is being run. Note that this operation is the only one that does not use tag parameters in col.2. It is recommended to run this as the 9th and last iteration in each script.

Parameters: *none*

STOP-SCRIPT

Stops the script from running. All other operations scheduled after this command are not performed.

Parameters: 1. subfield code and subfield contents for the condition (for example, \$\$a = *PUB*); can be left blank (see Appendix, entries 11 and 12).

41.4 Generic Fix Doc (p_file_08) Script Examples

1. Create tags 035 and UID from 001 (for Blackwell's MARC with Books files):

```
!-!!!!!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
2 001                                COPY-SYSTEM-NUMBER          035  ,L,a,(OrLoB)
2 001                                COPY-SYSTEM-NUMBER          UID   ,L,a,(OrLoB)
3 001                                DELETE-FIELD
```

2. Create tags 035 and UID from non-standard location of system number (LaserQuest):

```
!-!!!!!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
1 035##                              REPLACE-STRING          $$q,$$a(LQUEST)
1 035##                              COPY-FIELD             UID   ,L
```

3. Change tag 533 – Replace the string *Wash., D.C.* with *Washington, D.C.*
 The ^ sign (set in the first position of the operation parameters) defines the delimiter between the source text and the target text.

```
!-!!!!!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
1      533##                          REPLACE-STRING-GENERAL          ^Wash.,
D.C.^Washington, D.C.
```

4. Add a field to all records in a file (for example, 987 to indicate Pinyin processing):

```
!-!!!!!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
1 LDR  ##                              ADD-FIELD          987  ,L,$$aPINYIN$$bCaQMM$$c20001130
```

5. Change 400 to 800, correcting indicators and removing subfield w:

```
!-!!!!!!-!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 4002# CHANGE-FIRST-IND 2 1
1 400## CHANGE-SECOND-IND #
1 400## DELETE-SUBFIELD w
1 400## CHANGE-FIELD 800
```

6. Add hyphen to all ISSNs in 022 if not present as 5th character of the subfield:

```
!-!!!!!!-!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 022## EDIT-SUBFIELD-HYPHEN a,005
1 022## EDIT-SUBFIELD-HYPHEN y,005
1 022## EDIT-SUBFIELD-HYPHEN z,005
```

7. Extending 007 for computer files to new length with fill character:

```
!-!!!!!!-!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 007 c FIXED-FIELD-EXTEND 006,014,|
```

8. Using fixed field value changing operations with and without ranges of positions:

```
!-!!!!!!-!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 007 c 001 013 FIXED-CHANGE-VAL-RANGE -,|
1 007 h 012 FIXED-CHANGE-VAL b,i
1 008 SE 030 032 FIXED-CHANGE-VAL ###,^^^
```

9. Using fixed field range operations (for example, left-justify Nature of Cont. after deleting a value):

```
!-!!!!!!-!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 008 BK 024 027 FIXED-CHANGE-VAL-RANGE y,^
1 008 BK 024 027 FIXED-RANGE-OP LJ
```

10. Do not continue processing the record if the 010 field is not present:

```
!-!!!!!!-!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 010## COND-LOAD-VAL-FIELD N
```

11. Do not continue processing the record if the 010 field has a subfield \$z with the text DONTLOAD:

```
!-!!!!!!-!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 010## COND-LOAD-VAL-MATCH Y,$$zDONTLOAD
```

12. Using a STOP-SCRIPT operation to avoid the performance of scheduled operations when a field with specific contents is present:

```
!-!!!!!!-!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 OWN STOP-SCRIPT $$a = *PUB*
1 LDR ADD-FIELD OWN ,L,$$aXPUB
```

13. Using a STOP-SCRIPT operation to avoid the performance of scheduled operations when a specific field is present:

```
!-!!!!!!-!!-!!-!!-!!-!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 035 STOP-SCRIPT
```

42 Client Setup (catalog.ini)

The catalog.ini file defines settings for the Cataloging client. This chapter presents and explains the following sections of the catalog.ini file:

ConvertFile on page 126

DuplicateRecord on page 130

Editor on page 128

ExpandTemplate on page 130

Form on page 128

General on page 132

HolOwnTextDefaults on page 131

Locate on page 131

OffLine on page 131

RecordBar on page 132

RecordTree on page 132

[RfidMedia] on page 133

Scan on page 131

LOW on page 133

42.1 Catalog.ini Settings

Explanations on settings related to the Items functionality are documented in the Items chapter.

42.1.1 [ConvertFile]

[ConvertFile]

The ConvertFile section is used for the Import Records subfunction of the Cataloging module.

```
Convert1=L,Books in print conversion,LOCAL,BIP,0
```

The "convert" lines list the available conversion programs. In the example above, the "Books in print" string is the conversion name displayed in the Convert Procedure drop-down menu.

The definitions of the "convert" line include the type of conversion and, if needed, specific parameters (for example, character conversion).

The following is the basic structure for the "convert" lines:

```
ConvertN=TextALPHA,Text,ConversionType,ProgramPath,[Parameter1,Parameter2,...,ParameterN]
```

Text: Text that is displayed in the Convert Procedure drop-down menu.

TextALPHA: Alpha of the text.

ConversionType: Type of conversion. The conversion mechanism of the Import Records subfunction is based on two different types of conversion: conversions performed at the level of the client and remote conversions (that is, conversions performed by server routines). For conversions performed at the level of the client, this should be set to LOCAL. For remote conversions, this must be set to REMOTE.

ProgramPath: The path of the program to be executed. For remote conversions, this should always be set to REMOTE.

Parameters: Parameters for the conversion program (optional and program-dependent). For remote conversions, Parameter1 must match a conversion routine from the pc_tab_cat_conv table (column 1).

When converting external records into ALEPH format and importing them into your system, you can convert different codepages into UTF-8. To support this, you can add a parameter to the convert lines of the Convert section of the catalog.ini file. The parameter is used to define the input codepage that is to be converted into UTF-8. Following is the convert line for CDMARC records in Cyrillic:

```
Convert5=L,CDMARC conversion for Cyrillic,LOCAL,CDMARC,CP1251
```

This parameter is defined by defining CP + the codepage number (for example, CP1251 for Cyrillic).

If no conversion is needed, this parameter can be set to CPUTF.

```
DefaultInputDir=
```

You can use the DefaultInputDir variable to set the default directory that is opened when the user clicks the button at the right side of the Input File field from the Import Records subfunction. If you leave it blank, the default directory is set to the ConvertIn directory under the Catalog directory.

```
DefaultOutputDir=
```

You can use the DefaultOutputDir variable to define the default directory in which the converted files are stored. If you leave it blank, the default directory is set to the ConvertOut directory under the Catalog directory.

For more information on the Importing Records setup, refer to Importing Records on page 108.

42.1.2 [Form]

```
[Form]
FontSizeX=12
FontSizeY=20
```

The `FontSizeX` and `FontSizeY` lines are used to define the grid for the fonts of the cataloging forms.

42.1.3 [Editor]

```
TabCompletion=Y
```

If the `TabCompletion` flag is set to *Y*, then for subfields that have a list of options defined (`tag_text.dat`), it is possible to type the beginning of the text and press the `Tab` key so that the system fills in automatically the complete string.

```
AutoSaveTimeout=1
```

The `AutoSaveTimeout` variable is used to define the interval - in minutes - between autosaves of local records. If the variable is set to *0* (zero), the records are not saved automatically.

```
UseOldSystemNumber=N
```

The `UseOldSystemNumber` flag is used to define whether the system number of a record that is being duplicated should be kept as the system number of the new copy of the record. This flag should be set to *N*.

```
DisplayTagInfo=Y
```

The `DisplayTagInfo` flag determines whether or not the catalog name tags are displayed in the catalog draft in addition to the (usually numeric) field tags. If the flag is set to *Y*, the name tags are displayed.

```
HighLightTag=Y
```

The `HighLightTag` flag determines whether the tag of the field that is currently being edited appears highlighted or not. If the flag is set to *Y*, the tag is highlighted while the field is edited.

```
EditTag=Y
```

The `EditTag` flag determines whether the code tag can be edited/changed or not. If the flag is set to *N*, the cataloger will not be able to overwrite tags.

ExpandNewTag=Y

The ExpandNewTag flag determines whether the subfields defined in the marc_exp.dat are displayed when a field is selected from the list of valid fields - available by using the hotkey F5 or by selecting the New field (choose from list) option from the Edit menu -.

SortDeleteEmptyFields=Y

The SortDeleteEmptyFields determines whether or not empty fields are deleted when the Sort record option is selected from the Edit menu.

FontSizeX=10

FontSizeY=17

The FontSizeX and FontSizeY variables are used to define the grid for the fonts of the cataloging draft (for example, tags, indicators, contents).

BackColor=255,255,255
InfoColor=128,000,000
TagColor=000,000,255
IndColor=000,000,255
SubColor=192,000,000
FieldColor=000,000,000
DeniedFieldColor=128,128,128
DeniedFieldBackColor=255,255,255

SelectForegroundColor=255,255,255
SelectBackColor=000,000,128
TagHighLightColor=255,255,255
TagHighLightBackColor=128,000,000
FieldColor1=000,000,000
BackColor1=000,255,000
FieldColor2=000,000,000
BackColor2=255,000,000

The above variables are used to define the colors of the different elements of the cataloging draft (for example, the color of highlighted tags, the color of the indicators).

DeleteTempDocumentsInterval=7

The DeleteTempDocumentsInterval variable is used to define the interval (in days) for NEW* records that have not been updated/created to be deleted automatically from the local drive. If the variable is set to 0 (zero), records will not be deleted automatically.

ShowUnicodeValue=Y

If the ShowUnicodeValue flag is set to *Y*, when placing the mouse pointer over a character in the catalog record - after about two seconds - a ToolTip appears above the character displaying the character's Unicode value identified by the hexadecimal representation of its Unicode number prefixed with a U, for example, U+0041 for "A". If the flag is set to *N*, this ToolTip is not displayed.

RemoteRecordUpdate=Y

The RemoteRecordUpdate flag determines whether the system displays a message asking whether or not to update the record in the Remote Catalog whenever the record is saved to the server. The default value is N.

RightClickMenu=EditActions

There are two GUI cataloging Editor menus: Edit Text and Edit Actions.

This flag defines the activation hot keys for those menus.

Possible values: EditActions or EditText

- **EditActions** – (default) Right click to activate the Edit Actions menu. Press shift+right click to activate the Edit Text menu.
- **EditText** – Right click to activate the Edit Text menu. Press shift+right click to activate the Edit Actions menu.

42.1.4 [ExpandTemplate]

[ExpandTemplate]

BK=..\template\temp_bk.mrc

This section can be used to define a default template for a specific record format. In this case, the default template is selected automatically by the system when the cataloger uses the Expand from template option for a record with the defined format. In the above example, the tem_bk.mrc template - located in the CATALOG/TEMPLATE directory - has been defined as the default template for records of BK (book) format. If no default template is defined, a pop-up dialog box is displayed for the user to select the appropriate template.

42.1.5 [DuplicateRecord]

[DuplicateRecord]

Library=ALL

This variable is used to define the library/libraries options when duplicating a record. Values are:

HOME - The record is duplicated automatically to the Home Library (this is the library to which the user is currently connected).

ALL - A window listing all libraries defined in the CATALOG/PERLIB.INI file is displayed allowing the user to select the library in which he wants the new record to be saved.

<library code > [, <library code>] - To define specific libraries for selection (for example, USM01, USM10, USM30).

42.1.6 [OffLine]

[OffLine]
OffLine=N

The OffLine flag determines whether or not the Cataloging module will work with a server connection. If the flag is set to *N*, the client connects automatically to the server when opening the module. If the flag is set to *Y*, no connection is launched and the cataloger can continue working in Offline mode. When working in Offline mode, the user has access to data that has already been downloaded to the local PC (for example, help screens, forms, and so on), but he will not be able to perform functions related to the server (for example, checking procedures, database update, and so on).

42.1.7 [Locate]

[Locate]
MergeRecord=Q

The MergeRecord variable specifies whether the located similar record should be merged automatically with the current record. If the variable is set to *Y*, then the selected similar record is merged automatically with the current record without a message being displayed. If the variable is set to *N*, then the catalog draft of the selected similar record is displayed. If the variable is set to *Q*, a message is displayed asking the user if the records should be merged.

42.1.8 [Scan]

[Scan]
IncludeAUTData=Y

The IncludeAUTData flag is used to determine whether or not additional authority information from the authority record should be displayed in the headings list of the bibliographic library together with the linked authority record. If the flag is set to 'Y', then the 260 (Complex See Reference - Subject), 664 (Complex See Reference - Name), 666 (General Explanatory Reference - Name), and 680 (Public General Note) fields from the authority record are displayed together with the linked bibliographic heading.

42.1.9 [HolOwnTextDefaults]

[HolOwnTextDefaults]

The OWN field is a special ALEPH field that can be used in two different ways:

It can be used to control update access to all types of MARC records (BIB, HOL, ADM, AUT). The user is checked for access/update permission according to the contents of the record's OWN field(s).

It can be used in holdings records to define the "owner" of the record, in other words, the sublibrary to which the record belongs.

This section of the catalog.ini file is used to define in which way the OWN field is used.

Activate=Y

If the Activate flag is set to Y, when holdings records are created the Enter Owner Information window is displayed. This window enables you to define the owner of the holdings record. If the flag is set to N, then this window is not displayed and users can continue using the OWN field to control update access.

SubLibrary=
Note=

The SubLibrary and Note parameters can be used to save default owner values. These parameters are automatically filled in by the system after clicking the Save Defaults button from the Enter Owner Information window.

42.1.10[General]

[General]
HOLItemSupport=N

The HOLItemSupport flag determines whether or not the installation supports HOL Items. In standard applications, this flag should be set to N.

LOWSupport=N

The LOWSupport flag determines whether or not the installation supports Local Owner functionality (Central-Local Cataloging). In standard applications, this flag should be set to N.

42.1.11[RecordBar]

[RecordBar]
FgColorDescript=000,000,255

The FgColorDescript instance determines the color of the text displayed in the Catalog bar.

42.1.12[RecordTree]

[RecordTree]
BkColor=255,255,255
Z=370
ShowDetailedInfo=N

BkColor=

The BkColor instance determines the color of the Cataloging tab. Note that all other lines under RecordTree should not be modified.

ShowDetailedInfo=N

ShowDetailedInfo determines whether or not (NEWnnn.MRC) will be presented for every opened record in the record tree or detailed information that includes title, author, system number, format and/or year.

When ShowDetailedInfo=N, the default (NEWnnn.MRC) will appear.

When ShowDetailedInfo=Y, a detailed information will be presented in the record tree.

To enable this, set message number 2004 in ./aleph/error_eng/pc_cat_c0203.

Here is an example:

```
!!!!-!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
! Tree node text of record - $1=title, $2=author, $3=system number,
$4=FMT, $5=Year
2004 0000 L $1[$2]
```

42.1.13[RfidMedia]

```
[RfidMedia]
ActivateReader=Y
SuccessMessage=Y
```

The [RfidMedia] section is required when library items are RFID-tagged. For more information on using RFID equipment, please refer to the How-to document in the Ex Libris Documentation Center that is relevant to your RFID vendor (for example, for Bibliotheca, you would refer to *How to set up a BiblioChip® interface in ALEPH® 500 - 18.01*).

ActivateReader=Y

This variable determines whether or not the RFID Reader is updated when the relevant GUI actions are triggered.

SuccessMessage=Y

This variable determines whether or not a message indicating success is issued when the RFID Reader update action succeeds.

42.1.14[LOW]

```
[LOW]
DefaultOutputDir= C:\Temp
```

Define the output directory for the "List of Local Owners" window.
Default: C:\Temp.

43 Cataloging Tables

43.1 Library Tables

check_doc

The `check_doc` table lists all the checking programs that are run when the user chooses the **Check Record** option from the Edit menu or clicks the "Check Record" icon.

check_doc_doc

The `check_doc_doc` table defines field occurrences and dependencies between fields.

check_doc_field_xxx

The `check_doc_field_xxx` tables are used to define valid values for fixed-length fields. For example, the `check_doc_field_008` table is used to define valid MARC 21 values for the MARC 21 008 field.

check_doc_line

The `check_doc_line` table is used when performing tag specific validity checks on a field. The program checks:

- Validity of indicators and subfields.
- Repeatability and non-repeatability of subfields.
- The presence of mandatory subfields.
- Dependencies between subfields.

check_doc_line_contents

The `check_doc_line_contents` table is used to validate the contents of a field (for example, the ISSN).

check_doc.lng

The `check_doc.lng` table provides validation messages for the check doc programs.

check_doc_mandatory

The `check_doc_mandatory` table is used to define that certain check programs activate triggers or are defined as forbidden. Forbidden errors cannot be overridden and the user is unable to save the record.

check_doc_new_acc

This table defines the fields that should be ignored for purposes of the check messages regarding new acc headings. Up to 1000 codes that should be ignored can be defined. # can be used as a wild card.

check_doc_new_acc_aut

This table defines the fields that should be ignored when checking for new ACC headings, combined with a check in the relevant authority library.

check_doc_tag_text

The `check_doc_tag_text` table validates pre-defined texts for fields.

check_doc_unique_index

The `check_doc_unique_index` table is used to define the field that should be ignored when the system checks whether or not a duplicate record is opened in the Direct (Z11) index.

codes.lng

The `codes.lng` table defines the valid tags and aliases for the database.

fix_doc.lng

The `fix_doc.lng` table contains the text that appears next to fix routines when they are run manually from the Edit menu of the module. The table also determines whether the fix routine appears under the Fix Record option or under the Derive New Record option.

formats.lng

The `formats.lng` table defines the record formats codes (2 characters).

marc_country_codes

The `marc_country_codes` table in the `alephe/tab` directory is used to define the list of valid marc country codes. This table is used by the `check_val_country` that verifies that the position range of a given fixed-length field forms a valid country code (for example positions 15-17 of the 008 MARC 21 field).

marc_exp.dat

The `marc_exp.dat` table is used to define default subfields. The subfields defined are displayed in the following circumstances:

When a field is selected from the list of valid fields.

When the Open form option from the Edit menu is chosen for a field for which no form is available.

marc_language_codes

The `marc_language_codes` table in the `alephe/tab` directory is used to define the list of valid marc language codes. This table is used by the `check_val_language` that verifies that the position range of a given fixed-length field forms a valid language code (for example, positions 35-37 of the 008 MARC 21 field).

permission.dat

The `permission.dat` table defines allowed and denied tags for different catalogers.

scancode.dat

The `scancode.dat` table defines the heading lists that are used when the cataloger chooses one of the Search Headings functions.

tab00.lng

The `tab00.lng` table defines the system index files. There should be one such table for each language defined.

tab01.lng

The `tab01.lng` table contains the tag codes and names of MARC and ALEPH fields.

tab02

The `tab02` table defines text that is used by the `fix_doc_non_filing_ind` program. The program sets the value of a field's non-filing indicator. Fix programs are defined in the `tab_fix` table.

tab04

The `tab04` table converts one set of cataloging tags to another. Different conversion routines can be defined and linked to the `fix_doc_tab04_(01_99)` program. This can be used when importing records from a database with a different cataloging system.

Note

All tags not defined in this table are deleted from the record when activating the fix routine.

tab05.lng

The `tab05.lng` table defines captions for links between records using subfields in the LKR field.

In the LKR tag, the MARC tag defining the reason for linking two records is registered in subfield 'r'. `tab05.lng` defines the caption to display in the OPAC before subfields \$\$n and \$\$m.

tab11_acc

The `tab11_acc` table is used to assign fields to headings indexes.

tab11_aut

The `tab11_aut` table is used to define the headings files that the system uses to create hypertext links to FIND and BROWSE from the authority record. This allows the user to navigate the bibliographic database using the authority record fields.

tab11_ind

The `tab11_ind` table is used to assign fields to direct indexes.

tab11_word

The `tab11_word` table is used to assign fields to word indexes.

tab_aut

The `tab_aut` table establishes which headings indexes in the bibliographic database should be subject to authority control. This table also designates per ACC index which authority database should be checked for a match.

tab_bib_aut_match

Defines 6XX tags and AUT Index codes for the Create Additional Subject Heading(s) from Authority (manage-46)batch service.

tab_cat_hidden_fields

Defines which fields are not displayed in the Catalog Editor. Since the fields included in this table are not displayed, they cannot be updated through the Cataloging module.

tab_fix_notes

Defines texts used for translation in the `fix_doc_notes` fix routine.

tab_loader

`tab_loader` is located in the administrative library's `/tab/` directory (`./xxx50/tab`) and is used by the following services:

- Advanced Generic Vendor Records Loader (file-90)
- Load OCLC Records (file-93)
- Load MARCIVE Records (file-99)
- OCLC server

It defines processing regarding the creation of the holding records, items, orders, budget transactions, and load information.

tab_loader_def

tab_loader_def is located in the administrative library's /tab/ directory (./xxx50/tab) and is used by the following services:

- Advanced Generic Vendor Records Loader (file-90)
- Load OCLC Records (file-93)
- Load MARCIVE Records (file-99)
- OCLC server

It includes default values for fields in the (Z30), orders (Z68), and budget transaction records (Z601) that are created using this service.

tab_own

The tab_own table assigns the group of OWN values of a cataloging record (BIB, AUT, ADM or HOL) that are allowed for a particular OWN authorization.

tab_fix

Fix routines are standard library-defined procedures that automatically "fix" or make changes to cataloging records. The tab_fix table defines three aspects:

- The fix program that defines the type of "change" performed on the cataloging record.
- The fix routine in which the fix program runs.
- If required, additional parameters for the fix program.

tab_locate

The tab_locate table defines the locate routine to be used when searching for a similar record in other databases. Multiple lines can be set up for one library, in which case ALL lines are taken with an AND condition between them. The tab_locate table must include both the source and the target library.

tab_match

This table is used to specify the match routines performed by the Check Input File Against Database (manage-36) service and by the check_doc_match checking routine.

tab_match_acc

The tab_match_acc table is a sample table used to define the fields in the records to be checked against the headings index when the match_doc_acc program is used in the tab_match table.

tab_merge

The tab_merge table lists the merge routines which can be used by the fix_doc_merge program to merge or overlay cataloging records. Column 3 of the tab_fix table is used to define the merging routine that matches the relevant section in the tab_merge table.

tab_merge_overlay

The tab_merge_overlay table defines the fields to be retained, when overlaying cataloging records.

tab_move_record

This table is used to define the moving routines that are performed when records are moved through the Overview Tree in the Cataloging module.

tab_pinyin

This table is consulted to determine the fields on which `fix_doc_add_pinyin_check_sub9` and `fix_doc_add_pinyin_insert_sub9` routines will run. The `fix_doc_add_pinyin` programs run on the fields defined, if the content is CJK.

tab_publish

This table contains the specifications for extracting ALEPH records for publishing purposes. The table must be located under the `tab` directory of the library that contains the records to be extracted (in most cases this is the bibliographic and/or the authority libraries).

tab_subfield_punctuation

The `tab_subfield_punctuation` table is used to define subfield punctuation for fields. Punctuation for fields is necessary when the system automatically updates the bibliographic record from a linked authority record.

tab_z103

Column 1 of this `tab_z103` table defines which program runs for the building of links between records. There is an option to define special arguments (like SUDOC) in column 2.

tab_z105

The `tab_z105` table defines messaging between libraries. For example, the update of an authority record should cause an update of a `z01` (heading) in the bibliographic library.

tag_text.dat

The `tag_text.dat` table defines fixed values for specific subfields.

tagonnew.dat

The `tagonnew.dat` table defines the default fields when a new record is created.

44 Setting Up the LKR Field

You can control the display of the LKR field by making use of the `tab_fix_z103` table located in the bibliographic library's `tab` directory.

44.1 tab_fix_z103

The following are the available routines:

GUI-FULL - enables a sorted display in the Full+link tab of the Show node in the Search tab.

GUI-PRINT - enables a sorted display of holdings in printouts.

GUI-TREE - enables a sorted display of holdings in the navigation tree.

HOL-LIST - enables a sorted display in the list of Holding Records in the Cataloging module.

LOCAL-NOTE - allows proper base filtering (expand_doc_bib_local_notes and consults tab_fix_z103 in order to display the tags properly).

WEB-FULL - enables a sorted display in the *Full View of Record* window in the Web OPAC.

WEB-SHORT - enables a sorted display in the *Brief View of Record* window in the Web OPAC.

WEB-SET and WEB-Z103 - enables a sorted display in the *Create set of down-linked records* of the web OPAC.

Following is a sample of the tab_fix_z103 table:

```
! 1                2                3
!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>

GUI-FULL    fix_z103_sort_existing_key
WEB-FULL    fix_z103_sort_existing_key
WEB-SHORT   fix_z103_sort_doc_no
```

Key to the tab_fix_z103 table:

Column 1 - routine name

See the routines above

Column 2 - Program name

This is the name of the program that will perform a particular display. The available programs are:

fix_z103_filter_base

The program uses tab_expand_local_notes.conf to filter by base. It can use an expand routine in column 3, for example:

```
WEB-FULL fix_z103_filter_base
FILE=tab_fix_z103_local_notes.conf,EXPAND=WEB-FULL
```

fix_z103_filter_suppress

The program filters linked documents that are suppressed.

fix_z103_sort_852_b

The program enables the sort by 852 in the printed document.

fix_z103_sort_852_b_item_attr

The program acts like fix_z103_sort_852_b but also looks at tab_attr_sub_library type "7" (defines which sublibrary will be the first when sorting items list by sorting routine 06 - preferred sublibrary by IP).

fix_z103_sort_base

The program acts like fix_z103_filter_base, but uses the

tab_fix_z103_local_notes.conf file to control the z103 sort order.

fix_z103_sort_by_my_own

This program sorts records according to the value of the OWN field. All records with OWN tags assigned to the user according to tab_own are positioned at the beginning of the record list.

fix_z103_sort_doc_no

The program enables the sort by z103 doc number.

fix_z103_sort_existing_key

The program enables the sort of Z103 according to z103_sort. The z103_sort is depended on the \$s subfield of the LKR field.

fix_z103_sort_lkr_doc_no

The program enables the sort by z103 LKR doc number

Column 3 - Program arguments Contain programs additional information, such as table names. This column is used to define additional parameters for the programs.

45 Supporting additional filters in LKR Field

It is possible to use additional filters in the LKR field to filter items using all the levels of numeration and chronology.

The following are the additional LKR subfields:

- \$\$d - Fourth level of enumeration
- \$\$e - Fifth level of enumeration
- \$\$f - Sixth level of enumeration
- \$\$g - Alternative first level of enumeration
- \$\$h - Alternative second level of enumeration
- \$\$j - Second level of chronology
- \$\$w - Third level of chronology
- \$\$o - Forth level of chronology
- \$\$q - Alternative chronology.

To support the additional filters:

1. Make sure that Z103X table is defined in file_list of the Bibliographic and Administrative libraries.
2. If required, create Z103X table using util/o.
3. Set update_z103_lkr_extended program in tab_z103 of the Bibliographic and Administrative libraries.

For example, ./usm01/tab/tab_z103:

!	1	2
!!-!!->		

4. Run manage-12 to re-create the links.

46 LKR Updating Upon Item Enumeration and Chronology Modification

You can set Aleph to update the LKR field of a BIB record upon the modification of the related item's enumeration or chronology information.

The LKR field of type ANA is used to link bibliographic records in order to create a hierarchy between BIB records. This structure supports, for example, maintaining the following three levels of linking between BIB records:

- journal title
- issue title
- article title

The items are created for the journal title. The issue title and article title are linked to the journal title's items via the LKR ANA mechanism.

The LKR field is updated when the item enumeration or chronology information is modified using the following sections of the Aleph interface:

- Item Form – Serial Level tab
- Cataloging module – Binding
- Batch service: Update Item Records (manage-62)

For example, there is a bibliographic record with BIB record number 000057009 and Journal Title: Journal of Modern Art. There are 12 items record for this title, one per month (January 2010, February 2010, etc.).

There is a second bibliographic record with BIB Record number 000057010 and Issue Title: Journal of Modern Art – January 2010.

Record 000057010 is linked to record 000057009 in the LKR field:

LKR	a	ANA
	b	000057009
	n	Journal of Modern Art.
	y	2010
	v	14
	i	1
	j	Jan
	m	Journal of Modern Art

At the end of the year, the library staff binds all 12 items into a single bound volume, named Vol.14 Jan-Dec 2010. This single bound item replaces the 12 individual items.

After setting up this feature, the system automatically adds information to the LKR field of BIB Record 000057010 that points to the new bound item.

LKR	a	ANA
	b	000057009
	i	1-12
	j	Jan-dec
	m	Journal of Modern Art
	n	Journal of Modern Art.
	v	14
	y	2010

All other Issue Title BIB records (February, March, etc) are also updated with the new LKR field information.

To configure the LKR field updating:

1. Set PERIODIC-INDEX message type to “i” in tab_z105 of the ADM library.
2. In Column 3 (target library), set the BIB library in which to update the LKR field:

For example, ./usm50/tab/tab_z105

!	1	2	3	4	5	6	7	8	9	10
!!!!!!!!!!!!!!!!!!!!-!	!!!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!									
PERIODIC-INDEX	i	USM01								

3. In z105_library (for example, USR00), re-activate util E/11 (the message between libraries daemon).

This causes the system to find the bibliographic record that is attached to the LKR field and update the LKR field to reflect the latest enumeration and chronology information set in the item record.

For each bibliographic record that is identified, the following occurs:

- A new LKR field is created, containing filters according to the updated item’s information.
- If an identical LKR field already exists, the record is not updated in order to avoid multiple identical LKR fields.
- Previously existing LKR fields are not be deleted from the bibliographic record. There are often several copies of an item and only some of the copies are updated. This means that there are still items linked to the previous LKR fields. The previous LKR fields remain in the record to link to the remaining items that are not updated.
- Previously existing LKR fields remain even if there are no additional items connected to them. This does not cause broken links and is invisible to the end user.

Note that the LKR filters are applied for all LKR subfields only if Z103X (the extended linkage mechanism between documents) is activated. If Z103X is not

activated, the following subfields are not taken into account when linking the bibliographic record to specific items: LKR subfields d, e, f, g, h, w, o, and q.

47 tab100-related Entries in Cataloging

tab100 is the central configuration table for system-level, server-level and library-level variables. A few lines of the table are shown below:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!
HOLD-REQUEST-ITM-STATUS=Y
HOLD-REQUEST-COLLECTION=Y
CREATE-852-HOL=Y
CREATE-Z36H=Y
CREATE-Z37H=Y
CREATE-Z30H=Y
CHECK-INVOICE-CURRENCY=N
X852-ITEM-OVERRIDE=Y
HOLD-REQ-PROCESS-STATUS=N
RETURN-DURING-LOAN=0
CHECK-ORDER-BUDGET=Y
CHECK-UNIQUE-NAME-BIRTH=Y
ZERO-FINE-HANDLING=N
CHECK-ORDER-ISBN-ISSN=N
CREATE-ITM-FORM-ORDER-M=Y
BARCODE-DELETE-SPACES=Y
BOR-EXPIRY-DUE-DATE=Y
OVERDUE-LETTER-STYLE=Y
Z30-PRICE-FROM-ORDER=N
OVERDUE-LETTER-NO=1
CHECK-BARCODE=Y
MARC-TYPE=1
UNION-IGNORE-MATCH=deleted,circ-created
ADM-OWN-CHECK=N
```

ADM-OWN-CHECK

Defines whether or not the OWN field in the bibliographic record will be used to distinguish between ADM libraries in a multi-ADM environment.

Possible values are: Y or N

Y = the OWN field in the BIB record is used to distinguish between ADM libraries in a multi-ADM environment. When BIB records are pushed to an ADM library, the current staff user OWN permission is checked against the pushed BIB record.

N= No check on the OWN field from the BIB library.

The default value is N.

CREATE-852-HOL

Defines whether or not 852 subfields from the call number field in the BIB record will be automatically generated. This variable is applicable only to the HOL library (USM60).

Possible values are: Y or N.

Y = automatic generation of 852 subfields from call number fields in the BIB record (099, 098, 090, 092, 096, 050, 055, 060, 070, 082, 086)

The default value is N.

CREATE-Z00H

Defines whether a deleted BIB record is to be transferred to a history file for statistical purposes.

Possible values are: Y or N.

Y = transfer a deleted BIB record to a history file This is for statistical purposes only, and does not imply that there is capability to restore.

The default value is N.

CREATE-Z00R

Determines whether or not to create a Z00R record for each Z00 record.

Possible values are: Y or N.

Y = create a Z00R record for each Z00 record. Suitable for BIB, HOL and AUT libraries, but not for an ADM library.

The default value is N.

CREATE-Z106

Determines whether a Z106 record will be created automatically each time a cataloging record is created or updated.

Possible values are: Y or N.

N = record updates will not automatically generate Z106 records. In this case, the Z106 records can be created by running the Create/Update Z106 Table for "CAT" Field (p_manage_19) service available from the Catalog Maintenance Procedures option of the Services menu in the Cataloging module.

Y = each time a record is created or updated a Z106 record will be created.

The default value is N.

DOC-BLANK-CHAR

Defines what sign will be used to denote a blank in MARC 21 fixed fields. This should not be confused with the fill character |

Possible values are: ^ or -.

The default value is ^.

FORCE-USE-Z07

Determines how the system behaves when a Z07 record in an ADM or HOL library which is linked to a record in a BIB library is updated.

Possible values are: Y or N.

Y = A Z07 record will be created in the library (for example, ADM or HOL) although the document being updated does NOT belong to it. For example, a Z07 will be created in an ADM library when a record in a BIB library to which it is linked is updated).

The default value is N.

HOL-008-LNG

The HOL-008-LNG variable is applicable only to the holdings library (for example, USM60) and it is used to determine the default language code for the MARC 21 008 field in holdings records:

If the variable is set to 0, then the language code of the 008 field of the holdings record is set to the defaults specified in the tab_tag_text table.

If the variable is set to 1, the language code of the 008 field of the holdings record is taken from the bibliographic record based on standard system rules (that is, 008, 041, and so on).

The default is 1.

INDEX-ITM-LINK

The ITM link creates a link from one BIB record (for example, record A) to the items that belong to another BIB record (for example record B). If this flag is set to Y then when the items that belong to record B are changed, record A as well as record B will be re-indexed (that is, a Z07 record will be created for both).

For example, if you create logical bases by sublibrary, and the sublibrary of the item that belongs to record B is changed from MAIN to LAW, both records will be re-indexed and will appear under the LAW logical base.

Possible values are: Y or N.

Y = When an item is updated, BIB records linked to the item with an ITM link will be indexed. .

N = When an item is updated, BIB records linked to the item with an ITM link will be not be indexed.

The default value is N.

MARC-EXP-BLANK-CHAR

Determines whether or not to replace the tab100 variable DOC-BLANK-CHAR by a blank in non-fixed fields when exporting records in MARC format.

Y = The character defined by DOC-BLANK-CHAR will be replaced by a blank

N = The character defined by DOC-BLANK-CHAR will remain as is.

MARC-TYPE

Defines the type of the MARC record.

Possible values are: 1, 2, 3 or 4.

1=USMARC, 2=UNIMARC, 3=DANMARC, 4=MAB

The default value is 1.

OWN-FILTER

Determines whether or not the display filter is activated.

Possible values are: Y or N.

Y = The display filter based on `tab_own` is activated (only for HOL or BIB). N = The filter is not active.

The default value is N.

UNION-IGNORE-MATCH

If a value defined here is present in the \$\$a subfield of STA field in records, any related records will not be found as equivalent to other records.

Possible values are: deleted,circ-created

USE-ACC-TEXT

Determines how the system deals with GEN headings and AUT records when copying into a BIB record.

Possible values are: Y or N.

Y = using CTRL+F3/F4 in cataloging the system copies the contents of the chosen GEN heading into the BIB record.

N = using CTRL+F3/F4 in cataloging the system takes the preferred term (MARC = 1XX;MAB = TMP01) from the AUT record and copies it to the BIB record.

The default value is N.

Z01-TAG-SENSITIVE

Determines whether or not the Z01 record is tag-sensitive.

Possible values are: Y or N.

The default value is N.

EXPAND-ITEM-UPD-TIT

Determines whether or not the BIB's 245\$\$a is updated when activating the GUI-Cataloging-Expand from Item Barcode tool.

Possible values are: Y or N.

Y = The Expand from Item action replaces the 245\$a title field with the item's description (Z30-DESCRIPTION).

N = The Expand from Item action does not update the 245\$a title field.

The default value is Y.

48 Setup of ADM Libraries

Note that no ADM libraries should be defined for the **Connect to..** command in the ALEPH menu. In other words, this type of library should not be listed in the catalog/per_lib.ini file. These libraries are accessed from the related BIB library. There is no reason to access them directly.

In addition, note that the **Select ADM Library..** command in the ALEPH menu is used to specify the active ADM environment. This means that all ADM services should be listed in the menu-catalog.xml file and there is no need for the menu-catalog-adm.xml. All jobs that run under an ADM environment must include the <admin_library>Y</admin_library> tag, for example, p-item-03.xml.

49 Matching Records

ALEPH contains a number of matching programs which allow cataloging librarians to match cataloging records according to the matching routines defined in the tab_match table located in the library's tab directory.

Key to the tab_match table:

Column 1 - Match Code

This is the unique match routine code. Each routine performs a particular type of matching operation.

Column 2 - Matching Program

The following are the available match programs:

match_doc_uid: The matching is based on a direct index (Z11). The parameters column (Column 3) must contain either the index name (Column 5 in tab11_ind) or the tag code (Column 1 in tab11_ind). For example, if tab11_ind is defined as follows for the ISBN direct index:

```
1      2      3      4      5      6      7 8
!!!!-!!!!-!-!!!!!!!-!!!!-!!!!!!!-!!!!-!-!
020          ISBN  az
```

The parameters for a match based on the ISBN can be defined as follows:

```
XXX  match_doc_uid          I-ISBN
```

or

```
XXX  match_doc_uid          T-020
```

Use either I-<index code> or T-<tag code> When using T-<tag code>, there must be an exact match. If tab11_ind col.1 has 020##, this table must have T-020## as well.

match_doc_uid_2: Matching is based on a direct index (Z11). The parameters column (column 3) must contain the index name and the tag code as a unique value. This only works if they are the same (example tag 035 and index 035) in tab11_ind. For example, if tab11_ind is defined as follows for the 035 direct index:

```
1      2      3      4      5      6      7 8
!!!!-!!!!-!-!!!!!!!-!!!!-!!!!!!!-!!!!-!-!
035          035  az
```

then, the parameters for a match based on the 035 can be defined as follows:

```
XXX   match_doc_uid_2           035
```

match_doc_acc: Matching is based on a headings (ACC) index. The argument defined in Column 3 is a table name. This table lists the tags in the record to be checked against the headings index.

match_doc_script: Uses a table containing a special script for matching records. The table name is defined in Column 3.

match_doc_gen: Contains three sections to the program arguments: TYPE, TAG + SUBFIELD, and CODE. The ACC type can have an additional TRUNCATION argument.

- TYPE defines the search method for finding a match:
 - TYPE = SYS: searches against DB system number, which is expressed as CODE=001
 - TYPE = IND: searches against IND Z11 index
 - TYPE = ACC: searches against the filing text field of the ACC Z01 headings index

- TAG + SUBFIELD relates to incoming record only. The tag content is normalized using the same filing routine that is used for IND or ACC code.

You can define a specific subfield such as SUBFIELD=a or define matching on any subfield one by one with SUBFIELD=EACH. You can also define matching on any subfield "x" one by one with SUBFIELD=EACHx.

- CODE index name defines the code of the index that is searched in order to find a database record. TRUNCATION=Y can be added to the ACC match type. If this argument is present, the match is performed using a truncated search, that is, the incoming record's field is considered a match if it is contained within the heading.

Column 3 - Program Arguments

For match_doc_uid, this column contains the index code or the tag code used for the direct match.

For match_doc_acc this column contains the table name of the table that contains the tags to be checked against the headings index. Here is a section from the table

1	2	3
F96	match_doc_uid	I-ISBN
RLIN	match_doc_uid	T-020
OCLC	match_doc_script	tab_match_script_oclc
OCLC2	match_doc_uid_2	035
MRCV	match_doc_uid	T-909##
CAT	match_doc_uid	I-ISBN
CAT	match_doc_acc	tab_match_acc
MATCH	match_doc_script	tab_match_script.tst

When `match_doc_script` is used, a corresponding table must be defined. In the case of the OCLC match code in our example, the system uses values in the `tab_match_script_oclc` table:

```

!1      2              3      4              5
!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!>
01 match_doc_gen      1      goto 03      TYPE=IND, TAG=035##, CODE=035
01                    0+      goto 02

02 match_doc_gen      20-    goto 03      TYPE=ACC, TAG=245##,
SUBFIELD=abdefgknp, CO
DE=TIT, TRUNCATION=Y
02                    20+    stop

```

Key to the `tab_match_script_oclc` table:

This table contains five columns:

Column 1 - The match set identifier

Column 2 - The name of the match program.

In this example there are three match programs. More programs are available:

match_doc_gen: contains three sections for the program arguments: TYPE, TAG + SUBFIELD, and CODE. The ACC type can have an additional TRUNCATION argument.

TYPE defines the search method for finding a match:

TYPE = SYS: searches against DB system number, which is expressed as CODE=001

TYPE = IND: searches against IND Z11 index

TYPE = ACC: searches against the filing text field of the ACC Z01 headings index

TAG + SUBFIELD relates to incoming record only. The tag content is normalized using the same filing routine that is used for IND or ACC code.

CODE index name defines the code of the index that is searched in order to find the database record. TRUNCATION=Y can be added to the ACC match type. If this argument is present, the match is performed using a truncated search, that is, the incoming record's field is considered a match if it is contained within the heading.

match_doc_filter_hvd uses the program arguments SE-TABLE-NAME= and MO-TABLE-NAME=. The matching procedure uses the additional table(s) registered here for more specific matching arguments. The program automatically rejects all matches if the incoming record format is not SE or BK, and automatically rejects matches if there is a mismatch on the FMT field.

match_doc_filter_by_weights checks additional parameters existing in `tab_weights`. It can accept two types of Program arguments:

1) TABLE-NAME=<table name>

The following is an example taken from the `tab_match_script` table:

```
!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!-!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
06 match_doc_filter_by_weights      1      merge      TABLE-
NAME=tab_weights
```

2) `<table name>` - without the “TABLE-NAME=” words

The following is an example taken from the `tab_match_script` table:

```
!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!-!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
06 match_doc_filter_by_weights      1      merge      tab_weights
```

Column 3 - Number of Matched Records. Refers to the number of records in the database that match the incoming record. You can specify an exact number, an upper number limit (nn-) or a lower number limit (nn+). 0+ indicates at least one match; 0 indicates no match.

Column 4 - Action. Indicates the action to be taken where the condition of number of matched records is true. Supported actions are: skip (to skip to the next match set); stop (to stop script execution); goto `<xx>` (to jump forwards/backwards to a different match set `<xxx>`); `<any text>` acts in the same manner as skip. The table above uses the goto and the stop actions.

Column 5 - Arguments. Lists the match program arguments. For the 01 match set, the program arguments are: TYPE, TAG and CODE. For the 02 match set, the program uses the TYPE, TAG + SUBFIELD, CODE and TRUNCATION arguments.

50 Setting Up Services

50.1 Retrieve Catalog Records (ret-01)

You can enlarge the conditions list from two fields to three fields and from one subfield to two subfields within each field.

The third tag field and the second subfield for each tag field are hidden. In order to activate these two options, delete the `<hidden>` XML tag.

For example, the following argument is hidden:

```
<control>
  <hidden>
    <argname>F18</argname>
    <label>Subfield</label>
    <size>1</size>
  </hidden>
</control>
```

To activate it, remove the following two XML tags: `<hidden>` and `</hidden>`.

51 CJK Unicode Characters

There is an option in the Cataloging Record Editor to display and Edit CJK Extension A and Extension B characters. This option includes 4-character Unicode values (U+0000-U+FFFF) and 5-character Unicode values (U+10000-U+FFFFF).

CJK Extension A and Extension B characters can be displayed and entered into cataloging records using Unicode mode (F11):

For Extension B: enter '+' and then enter the 5-character Unicode value (for example, +20000).

For Extension A: act as usual. Enter the 4-character Unicode value (for example, 004C).

Note that CJK Extension A and Extension B characters can be used also in “Open Form” functionality.

ALEPH Keyboard and IME methods work with Extension A values only.

To see SuperCJK Extension A and Extension B characters, install the appropriate fonts (for example: zyksun).

The following lines should be added to `./Alephcom/Tab/Font.ini`:

```
EditorField      20000 30000 zyksun   Y N N 16 DEFAULT_CHARSET
ListBox##        20000 30000 zyksun   Y N N 16 DEFAULT_CHARSET
UnicodeEdit      20000 30000 zyksun   N N N 16 DEFAULT_CHARSET
```

52 Publishing

ALEPH publishing is a mechanism which allows sites to extract records from the ALEPH catalog for distributing purposes. (for example, for publishing to search engines and search tools such as Google and Primo).

The extract process has two different flows: initial and ongoing. The initial extract usually includes all records in the catalog, while the ongoing extract mainly deals with new and updated records.

Both publishing processes place the documents into the data repository which is a directory that is locally defined. The data repository consists of Z00p records.

Note that the extract process can be performed on the whole database or on a specific logical base. In addition, extracted records can be modified to include information added by standard ALEPH procedures such as FIX and EXPAND.

52.1 Initial Extract Process

The initial extract process is performed by running the Initial Publishing Process (publish-04). This service can be run from the Publishing submenu of the Services menu in the Cataloging module.

The selected range of records for the specified set will be extracted. (It is possible to choose ‘ALL’ for all the sets specified by the System Librarian).

Note that the service will not run if there is at least one published record in the given range. If you still want to run the service for the selected records, first use the Delete ALEPH Published Records (publish-05) service in order to delete the existing Z00P records in the selected range.

The extraction (initial and ongoing) is performed according to the tab_publish table located under the tab directory of the library that contains the records to be extracted (for example, XXX01).

Here is an explanation for the table's columns:

Column 1 – Publishing Set

This column contains the code of the set of records to be extracted. For example, if the database needs to be extracted in two separate formats for two separate publishing platforms (such as Google and Primo) then two separate sets should be defined in the table. Note that the code must be in upper case.

Column 2 – Base

A set can be the entire database or a section of the database as defined by a logical base. This column contains the code of the desired logical base from the tab_base.lng table. If the column is left blank, the entire database will be extracted for the set.

Column 3 – De-duplication (for future use)

This column is currently not in use.

Column 4 – Fix and Expand Code

This column contains the fix and expand code of the routines that should be applied before the record is extracted.

Note: To avoid conflicts in tab_fix, do not use the string FULL as a fix and expand code.

Column 5 – Repository Format

This column determines the format of the records in the repository. The supported formats are:

- MARC_XML
- MAB_XML
- HTML
- OAI_MARC21_XML
- OAI_DC_XML

52.2 Ongoing Extract Process

The ongoing extract process is required in order to reflect changes to the database such as the deletion of records and updates to the bibliographic records/holdings records/item records, etc. The ongoing extract process has two main stages:

- The trigger for the extract
- The export of changed/new records to the repository

The triggering mechanism for the extract is based on the ALEPH indexing trigger mechanism. In ALEPH a Z07 record is created for each new or modified record. For the ongoing extract process, when a Z07 record is created for a bibliographic record, the system creates a Z07P record. The Z07P is the trigger for the ongoing extract process.

Note that the creation of z07p is depended on whether tab_publish exists in the Bibliographic and/or Authority library. If the table doesn't exist, no z07p records will be created.

Z07 records are created for bibliographic records in various cases such as changes to the related holdings records, authority records, items, etc. This ensures that bibliographic records are indexed not only according to their own data but also according to associated data. Since the Z07P is based on the Z07, this guarantees that the extracted records, which might contain information derived from FIX and EXPAND procedures, are correctly updated.

Z07p records are also created when an item is loaned, returned, or when a hold request is placed on it.

The timing of the creation of the trigger record (Z07P) differs depending on whether or not the publishing set is created based on a logical base. If the publishing set is not base-dependent, the Z07P is created immediately after the creation of the Z07 record (before it is processed by the UE_01 indexing daemon). If the publishing set is base-sensitive, the record must be indexed before it is extracted. In this case, the Z07P record is created only after the Z07 record has been processed by the UE_01 daemon. The reason for this difference is that in sites where the publishing sets are not base-sensitive, there is no reason to wait for the indexing of the records in order to start the ongoing extract process.

Note that the timing of the creation of the Z07P records explained above is not dependent on the specific publishing set but depends on whether there is at least one entry in the tab_publish table that is base dependent. In other words, if there are four publishing sets defined in the table but only one is base sensitive, then in all cases the Z07P record will be created after the processing of the UE_01 daemon.

The handling of the changed repository records (z00p records) is performed by the Ongoing Publishing utility ue_21. This utility compares the record for which the Z07P was created with the z00p record in the repository. If the records differ (after EXPAND and FIX), the record is handled. When the service finishes processing the triggered documents, the Z07P records are deleted.

In order to prevent unnecessary update because of date change and update of a non important field, a line such as the following can be added to relevant menu in tab_fix:

```
! 1                2                3
!!!!!!-!!!!!!>
LAM   fix_doc_do_file_08                del_005
```

A file like del_005 should be located in the library's import directory under the tab directory. It can include fields that will always be updated in the bibliographic record such as field 005. These fields will be deleted and the z00p record will not be updated.

Here is an example for the syntax:

```
! 2 3 4 5 6 7                8                9
!-!!!!-!!-!!-!!-!!!!-!!!!!!-!!!!!!>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
1 005                DELETE-FIELD
```

The Ongoing Publishing utility, ue_21 should run on a regular basis in order to ensure that the repository is up to date. Ue_22 stops ue_21. The performance of ue_21 can be improved by setting the aleph_start/ prof_library variable: num_ue_21_processes. This variable enables you to divide the running of the job into several processes. The variable can be set in aleph_start or in the prof_library file of the publishing library. Setting the variable in \$alephe_root/aleph_start or aleph_start.private affects all of the publishing libraries. Setting the variable in \$data_root/prof_library of the publishing library affects only this library.

If ue_21 tries to upload an invalid xml (containing bibliographic information), a file will be written under \$data_scratch directory with the following name convention: util_e_21.xml_err.<YYYYMMDD>.<HHMMSSmm>. The file contains the document number and the library of the document which was not updated due to the invalid xml. The library should look every couple of days for these files and handle them.

The changes triggered by z07p update the z00p records. P-publish-06 service can take the updated z00p records based on dates and record numbers and create a tar file for them. This file can be later transferred to different publishing platforms.

Note that if a change is made to a base or to a base definition in tab_base.lng and this base exist in tab_publish, p-publish-05 and p-publish-04 should run to create initial load again. Ue_21 should be restarted.

52.3 Name Spacing in Publishing

The BIB library table, ./xxx01/tab/tab_md_ns_info, enables usage of additional or modified name spaces for the Aleph publishing platform.

This optional table can be used to define namespace information for provided formats.

Column 1 – the publishing set code of the set of records to be extracted

Column 2 – defines the namespace information for the publishing set

Example:

```

!           1           2
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!--!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!->
TEST1          xmlns="http://www.loc.gov/MARCxx/slim"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.loc.gov/MARC21/slim
http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd"

```

53 Upload BIB and Holding Information from Aleph to KERIS

Aleph uploads BIB documents and holdings from the local Aleph BIB library to KERIS using the Z39.50 protocol.

KERIS is the Union Catalog of the Korean Academic Libraries. The KERIS Server conforms to ANSI/NISO Z39.50-2003(reversion of Z39.50-1995).

KERIS Databases:

- UBIB – BIB documents including local holdings
- UHOL – Holdings linked to UBIB titles

Aleph supports the following update services:

- INSert for UBIB and UHOL
- MODify for UBIB

Aleph uploads to KERIS in two modes:

- Online Upload ("One by one" manual uploading via GUI-Cataloging module) – The user is able to upload new BIB documents, altered BIB documents and BIB documents that were deleted. This is available for both BIB documents that were downloaded from KERIS and new cataloged documents. The user is able to upload changes in the BIB document's holdings as well. This mode can be applied by using the "Remote" menu in the Cataloging module.
- Batch uploading – The user is able to upload group of BIB documents to KERIS using the batch service: Upload Remote Records via Z39.50 (print-30).

Before uploading BIB documents to KERIS, the documents are automatically enriched with additional information using the standard fix and expand mechanism.

53.1 Tables Set-Up Configuration

53.1.1 KERIS Z39.50 Gate Configuration

The KERIS remote database should be configured in Aleph as a Z39.50 gate.

The following are instructions for setting KERIS as Z39.50 base. In the sample below, the base code is "KERIS". (You may set the base code to any other 5 characters).

1. Configure the KERIS remote database as an external base of EXT01 library.
2. Set the following line in `./alephe/tab/tab_base.lng`

```
KERIS          KERIS          EXT01          EXT01
```

3. Set a line for KERIS in `./alephe/tab/z39_gate/z39_gate.conf`

```
include z39_gate_KERIS.conf
```

4. Set z39.50 gate configuration for KERIS:
`./alephe/tab/z39_gate/z39_gate_KERIS.conf`
5. Set the following lines in KERIS in order to upload to the KERIS z39.50 configuration file: `./alephe/gate/keris.conf` file:

```
Z58-ES-DELETE      es_delete_keris
Z58-ES-INSERT      es_insert_keris
Z58-ES-REPLACE     es_replace_keris
```

6. To enable the Remote menu in the Cataloging module, set the BIB library's `remote_catalog.dat` configuration table. The `remote_catalog.dat` file is located in the `./<BIB library>/pc_tab/catalog` directory.
7. Set the following line in `remote_catalog.dat` for the KERIS remote base:

```
!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!-!!!>
KERIS          KERIS          8
```

8. Run util M/7 in `bib_library` after updating `remote_catalog.dat`.
9. To send a repaired document to KERIS (via the "Remote" menu in Cataloging module), set the EXT01 library's `remote_catalog.dat` configuration table. The `remote_catalog.dat` file is located in `./ext01/pc_tab/catalog` directory.
10. Set the following line in `remote_catalog.dat` for the KERIS remote base:

```
!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!-!!!>
KERIS          KERIS          9
```

11. Run util M/7 in EXT01 library after updating `remote_catalog.dat`.

53.1.2 Expand and Fix Routines Setup

Before the document is sent to KERIS, additional information is added to the document which is required to update the KERIS database.

In order to enable document enrichment set `tab_fix` and `tab_expand` with the relevant `expand/fix` routines.

KERSU and KERSD are two special instances that are applied when the KERIS upload is performed. Those instances are set in column 1 of tab_expand and tab_fix. Relevant programs (expand and fix procedures) are defined in column 2 of those tables.

- **KERSU** routine is applied when uploading a record to KERIS (using the Remote function in the GUI Cataloging module or by submitting the batch "Upload remote records via Z39.50").
- **KERSD** routine is applied when uploading a deleted record to KERIS (using the "Remote" function in GUI Cataloging modules).

Set expand and fix routines in external virtual library (EXT01 in below sample setup), BIB MARC21 library (USM01 in the sample) and BIB KORMARC library (KOR01 in the sample).

Sample of Setup:

Sample Setup in EXT01 (external library)

Fix routines are defined in order to repair an external document and upload it to KERIS. In the following sample, the routine is keris_upload_fix.

tab fix:

```

./ext01/tab/tab_fix

      1              2              3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
KERSU fix_doc_do_file_08              keris_upload_fix

```

keris upload fix:

```

keris_upload_fix procedure should be set in:

./ext01/tab/import/keris_upload_fix (delete various fields)

!  2  3  4  5  6  7              8              9
!-!!!!-!!!-!!!-!!!-!!!!-!!!!-!!!!-!!!!-!!!!-!!!!-!!!!-!!!!-!!!!-!!!!>
1  SID##              DELETE-FIELD
1  CAT##              DELETE-FIELD
1  FMT##              DELETE-FIELD
1  SYS##              DELETE-FIELD
1  STA##              DELETE-FIELD
1  LOC##              DELETE-FIELD
1  OWN##              DELETE-FIELD
1  KER##              DELETE-FIELD

```

Sample Setup in USM01 (MARC21 BIB library):

Fix routines should be defined in order to upload updated document or delete document. In the following sample, the routines are `keris_upload_fix` and `keris_delete`.

tab fix:

```
./usm01/tab/tab_fix

      1              2              3
!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
KERSU fix_doc_do_file_08              keris_upload_fix
KERSU fix_doc_005
KERSU fix_doc_004_lkr
KERSD fix_doc_do_file_08              keris_delete
KERSD fix_doc_005
KERSD fix_doc_004_lkr
```

keris_upload_fix:

```
keris_upload_fix procedure should be set in:

./usm01/tab/import/keris_upload_fix (LDR position 10 set to "u" and various fields
are deleted):

!  2  3  4  5  6  7              8              9
!-!!!!!!-!!!-!!!-!!!-!!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 LDR          010 010          FIXED-CHANGE-VAL          #,u
1 SID##
1 CAT##
1 FMT##
1 SYS##
1 STA##
1 LOC##
1 OWN##
1 KER##
          DELETE-FIELD
          DELETE-FIELD
          DELETE-FIELD
          DELETE-FIELD
          DELETE-FIELD
          DELETE-FIELD
          DELETE-FIELD
```

keris_delete:

```
keris_delete procedure should be set in:

./usm01/tab/import/keris_delete (LDR position 5 is set to "d"):

!  2  3  4  5  6  7              8              9
!-!!!!!!-!!!-!!!-!!!-!!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 LDR          005 005          FIXED-CHANGE-VAL          #,d
```

tab expand:

Set expand routine to enrich the document with 852 item information. The Aleph sublibrary code should be converted to a code as required by KERIS. In the sample

below, the change_sub_library routine converts sublibrary codes ("LAW" and "MED") to KERIS codes (001 and 002 respectively).

```
./usm01/tab/tab_expand

!      1                      2                      3
!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
KERSU      expand_doc_bib_852_1
KERSU      fix_doc_do_file_08          change_sub_library
```

change sub library

```
change_sub_library procedure should be set in:

./usm01/tab/import/change_sub_library (converts Aleph sublibrary to Keris code):

!      2      3      4      5      6      7                      8                      9
!-!!!!!!-!!-!!-!!!!-!!!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>

1      852##                                REPLACE-STRING
$$bLAW,$$b001
1      852##                                REPLACE-STRING
$$bMED,$$b002
```

Sample Setup in KOR01 (KORMARC BIB LIBRARY):

Fix routines should be defined in order to upload updated document or delete documents. In the following sample, the routines are keris_upload_fix and keris_delete.

tab fix:

```
./kor01/tab/tab_fix

      1                      2                      3
!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
KERSU fix_doc_do_file_08          keris_upload_fix
KERSU fix_doc_005
KERSU fix_doc_004_lkr
KERSD fix_doc_do_file_08          keris_delete
KERSD fix_doc_005
KERSD fix_doc_004_lkr
```

keris upload fix:

```
keris_upload_fix procedure should be set in:

./kor01/tab/import/keris_upload_fix (LDR position 10 set to "c" and various fields
are deleted):

!      2      3      4      5      6      7                      8                      9
```

```
!-!!!!!!-!!!-!!!-!!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 LDR          010 010          FIXED-CHANGE-VAL          #,c
1 SID##
1 CAT##
1 FMT##
1 SYS##
1 STA##
1 LOC##
1 OWN##
1 KER##
1 KER##          DELETE-FIELD
```

keris delete:

```
keris_delete procedure should be set in:

./kor01/tab/import/keris_delete (LDR position 5 is set to "d")::

!  2  3  4  5  6  7  8  9
!-!!!!!!-!!!-!!!-!!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!>
1 LDR          005 005          FIXED-CHANGE-VAL          #,d
```

tab expand:

Set the expand routine to enrich the document with 852 item information. The Aleph sublibrary code should be converted to a code as required by KERIS. In the sample below, change_sub_library routine converts sublibrary codes ("LAW" and "MED") to KERIS codes (001 and 002 respectively).

```
./kor01/tab/tab_expand

!  1  2  3
!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
KERSU          expand_doc_bib_852_1
KERSU          fix_doc_do_file_08          change_sub_library
```

change sub library:

```
change_sub_library procedure should be set in:

./kor01/tab/import/change_sub_library (converts Aleph sublibrary to Keris code):

!  2  3  4  5  6  7  8  9
!-!!!!!!-!!!-!!!-!!!-!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!>

1  852##          REPLACE-STRING
$$bLAW,$$b001
1  852##          REPLACE-STRING
$$bMED,$$b002
```


53.1.3 Manual Upload Using the Remote Menu of Cataloging Module

The Remote menu of the Cataloging module uploads a single document to KERIS. It supports the following updates:

- Upload new and updated document
- Upload deleted document
- Upload suppressed document
- Send repaired document to KERIS

The upload action requires the following staff user privileges: Cataloging Record/Send record to remote server.

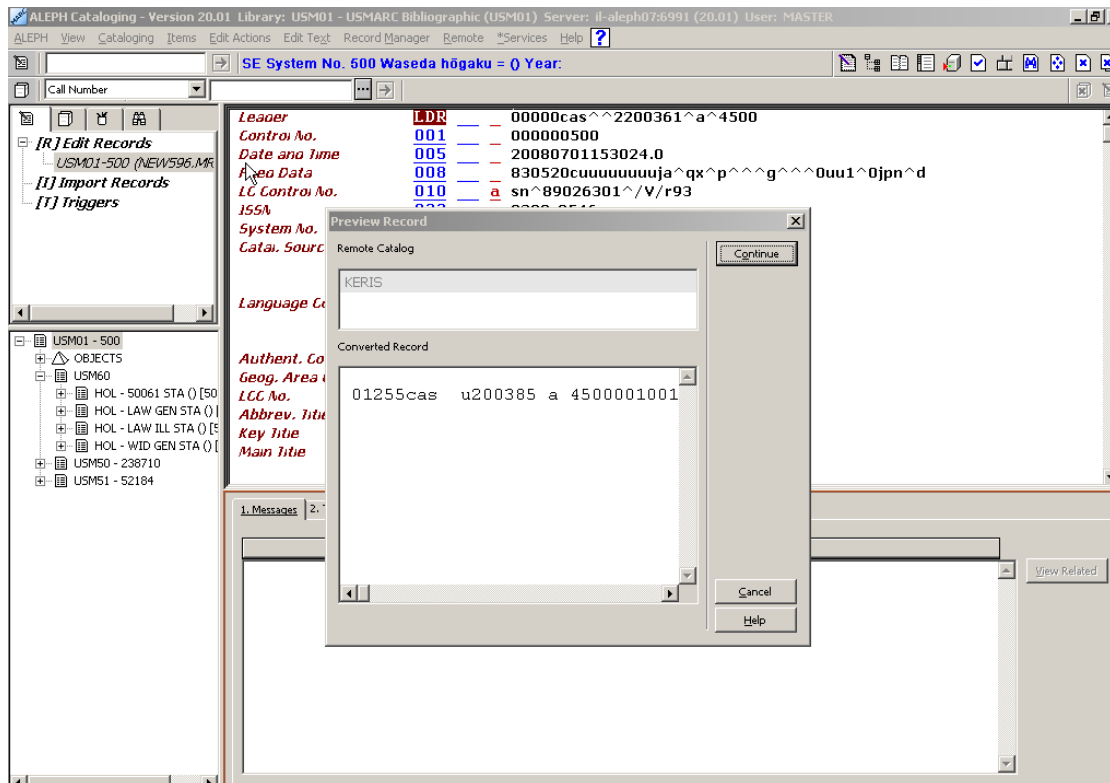
53.1.4 Upload New and Updated Document Manually

The users update BIB documents and holdings or create new BIB documents and holdings in the BIB library using the standard workflow supported by Aleph in the cataloging module. The insert and update operations on the BIB document are recorded in the CAT field on the document.

In order to update KERIS union catalog, the user should select the Add Record option in the Remote menu of the cataloging module.



Selecting the **Add Record** option opens the Preview Record dialog which enables the user to view the converted record that is sent to the remote system.



When the **Continue** button is clicked, the BIB document and its holding information is sent to KERIS database using the Z39.50 protocol.

Before sending the BIB document to KERIS, the BIB document is automatically enriched using the standard Aleph fix and expand routines, adding the holdings information and other necessary data.

The job action of the message to KERIS is always **INSert** and the database target in KERIS is determined using the 035 field on the document:

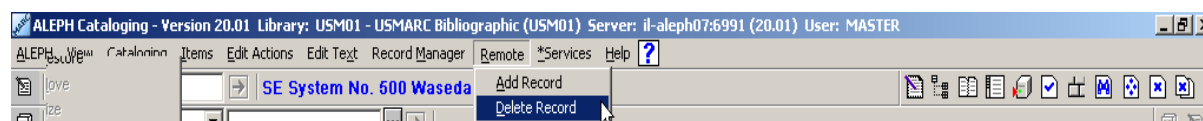
- BIB documents downloaded from KERIS includes 035 field with subfield \$\$a starting with "(KERIS)BIB" is uploaded to KERIS UHOL database (example of 035\$\$a: (KERIS)BIB00001075867).
- New BIB documents that were not downloaded from KERIS and do not includes 035 field with subfield \$\$a starting with "(KERIS)BIB" are uploaded to KERIS UBIB database.

After the BIB document is sent, the upload date is registered in KER field subfield \$\$a.

53.1.5 Upload Deleted Document Manually

Deleting the remote BIB document from KERIS must occur before deleting the BIB document from the local BIB library due to the fact that deleting the BIB document from the local library removes most of its data including its 035 field which identifies the source system of the document and the document number in that remote system.

Use the “Delete Record” option in the “Remote” menu in order to delete the BIB document from KERIS.



Select the **Delete Record** option in the Remote menu to send the enriched BIB document to KERIS using the Z39.50 protocol with the value “d” in the position 5 in the LDR field.

The job action of the message to KERIS is INSert and the database target in KERIS is determined using the 035 field on the document:

- BIB documents downloaded from KERIS and includes 035 field with subfield \$\$a starting with "(KERIS)BIB" is directed to KERIS UHOL database.
- New BIB documents that were not downloaded from KERIS and do not include the 035 field with subfield \$\$a starting with (KERIS)BIB but were uploaded to KERIS before are directed to the KERIS UBIB database.

The BIB document and its holdings are not deleted from the local BIB library automatically. After completing the delete from the KERIS union catalog, the user needs to delete the BIB document from the local BIB library using the standard delete functionality supported in Aleph in the cataloging module.

To delete the BIB document from the local BIB library, use the **Delete record from server (ctrl + R)** option in the “Edit Text” menu.

53.1.6 Upload Suppressed Document Manually

Adding the field STA with the value \$\$aSUPPRESSED to a BIB document and saving it to the local BIB library, cause it to be considered suppressed.

<i>Leader</i>	<u>LDR</u>	—	—	00656nas^^22002172^^4500
<i>Control No.</i>	<u>001</u>	—	—	000000222-4
<i>Date and time</i>	<u>005</u>	—	—	20020418155342.1
<i>Fixed Data</i>	<u>008</u>	—	—	770701c19049999ne^ p dut
<i>ISSN</i>	<u>022</u>	—	a	0922-6699
<i>Main title</i>	<u>245</u>	00	a	Jaarboekje voor geschiedenis en Oudheidkunde van Leiden en Omstreken.
<i>Varying title</i>	<u>246</u>	13	a	Leids jaarboekje
<i>Status</i>	<u>STA</u>	—	a	SUPPRESSED
<i>Imprint</i>	<u>260</u>	00	a	Leiden :
			b	<S.N.>.
			c	1904-
<i>Dates of Pub.</i>	<u>362</u>	0	a	jg.1- 1904-
<i>General Note</i>	<u>500</u>	—	a	Half-title: Leids jaarboekje.
<i>General Note</i>	<u>500</u>	—	a	Title varies slightly.
<i>General Note</i>	<u>500</u>	—	a	Issued by Vereeniging Oud-Leiden.
<i>Subject-Geo.Irm</i>	<u>651</u>	0	a	Leiden (Netherlands)
			x	History
			v	Periodicals.
<i>A.E. Corp. Name</i>	<u>710</u>	2	a	Vereeniging Oud-Leiden.

When the user suppresses the BIB document, he should upload the BIB document to KERIS as deleted BIB documents as described in section "Upload Deleted Document Manually"

When the user removes the STA field from the BIB document and saves the BIB document to the local BIB library, he should upload the BIB document to KERIS as described in section "Upload New and Updated Document Manually".

53.1.7 Upload Repaired Document Manually

When a BIB document is downloaded from KERIS, it is saved in a temporary EXT library. The cataloger cannot update and save the document in the EXT library since the EXT library is temporary.

The user can repair the downloaded BIB document in the EXT library via the cataloging module editor and upload it to the KERIS UBIB database using the **Update Record** option in the Remote menu of the cataloging module.



Note that the BIB document data is taken from the cataloging module editor instead of the Aleph database, as in the other remote actions, since the BIB document can not be saved.

The uploaded BIB document from the EXT library is sent to the KERIS UBIB database with a MODify action.

If the user wants to save the downloaded BIB document to Aleph, he should use the standard functionality applied as "Duplicate Record (Ctrl+N)" action under the Cataloging menu in the cataloging module.

53.1.8 Bulk Upload Using Batch Service

In addition to the manual upload to KERIS functionality, Aleph provides an option for bulk uploading to KERIS.

Bulk upload to KERIS functionality is divided into two stages:

- Initial filtering of BIB documents.
- Secondary filtering of BIB documents and uploads to KERIS via batch.

The purpose of the initial filtering is to create an input file of all updated and new records that should be uploaded to KERIS.

The initial filtering of BIB documents can be performed using the "Retrieve Catalog Records (RET-01)" batch process that enables retrieving documents according to various parameters, for example:

- Records that have items that were updated between dates ranges(From-To)
- System record number ranges(From-To)
- The last updated Cataloger

- The last updated Date (From-To)

The output BIB documents numbers is written to an output file that is saved in the \$alephe_scratch directory and is used as a basis for the secondary filtering.

The Secondary filtering of BIB documents and uploading to KERIS is done using the “Upload remote records via Z39.50 (print-30)” batch process.

The batch interface is accessed via Cataloging module-Services Menu-Retrieve Catalog Records- Upload remote records via Z39.50 (print-30).

The batch can be submitted by authorized staff users only.

Required staff user privilege: Record Services/Upload Remote Records Via Z39.50

The Batch parameters:

Input File: file name in \$alephe_scratch directory. This file needs to contain BIB documents numbers that should be uploaded to KERIS. Usually this file is the output file of “Retrieve Catalog Records (RET-01)” batch process.

Report Output File: The name of the file in which you want to save the output report.

Remote Base: Z39.50 base code to which the process sends the BIB document information.

Upload BIB Records: Secondary filter which allows the users to choose to upload:

- **New Records:** Records from the input file that were not uploaded to KERIS before.
- **Updated Records:** Records from the input file that were uploaded to KERIS before and have been updated.
- **All Records:** All the records that are ready for upload (new and updated) from the input file.

Using the CAT fields on the BIB document, it is possible to identify the last update date of the BIB document.

When the BIB document is uploaded to KERIS, subfield \$\$a in the KER field is updated with the last upload date.

Using these dates, it is possible to identify the BIB documents that are ready for KERIS upload and the BIB documents that are new to KERIS.

Sort By: The parameter by which the output report is sorted.

Report Format: The format of the output report IB Records (template file name: remote-base-upload.xsl).

1.3.1 The Batch Execution

Before sending the BIB documents to KERIS, the documents are enriched using the fix and expand standard functionality. In addition, the BIB documents LDR field are populated in position 10 according to the library documents format:

- Value “c” for KORMARC documents.
- Value “u” for MARC21 documents.

After the BIB document is sent, subfield \$\$a in the KER field is updated with the upload date, using a fix routine.

53.1.9 The Batch Output Reports

After the batch is submitted, a detailed output report is produced, describing the outcome of each uploaded BIB document. The Upload status can be one of the following:

- Success
- Library problem
- Document problem
- Base problem
- Not uploaded

The report template filename: remote-base-upload.xsl.

The template translation filename: remote-base-upload.trn.

In addition, print-30 batch service generates statistic information that can be retrieved using the TCO Batch Summary Report (sys-90). The summary includes the following statistics information:

- Number of documents processed

- Number of documents successfully uploaded to KERIS
- Number of documents that failed to upload to KERIS

54 OUF Loader

The OUF Loader (p_ouf_load) batch service loads into Aleph the bibliographic records BIB, AUT, and HOL fetched from an OCLC/PICA system.

The OUF Loader process receives input in Aleph sequential format. BIB, AUT, and HOL records have to be processed in separate files. Retrieving the records from the OCLC source system via OUF (OCLC Online Update Fetch) and converting the records into Aleph sequential format is done by external programs.

Possible actions (update modes) are add, update, delete, and linked. Both add and update do the same thing - they add as new record if it does not exist in database and updates it if it is found in database. The difference is that for add, a warning log is created if the record is already in the database, and for update a warning log is created if this record is not found. Linked value means the same as add or update but no warning log is created whether the record is new or not and there are more restricted conditions on the record timestamp.

Records identification is based on a OCLC ppn number, put in Marc 001 field and indexed as 001 IND index (Z11). Proper ppn and working unique 001 index are necessary to update the record and for the link to work properly. Before a record is inserted a fix routine can be applied. Before a record is updated a merge routine can be applied. In addition it is possible to apply fix routines before and after the merge. The OUF Loader is intended to be part of a regular batch process for updating the Aleph database with data from the OCLC union catalogue. Therefore, the OUF Loader cannot be run from the Services menu of the Cataloguing module, but only from a command line or other scripts.

54.1 Instructions for Running the OUF Loader

54.1.1 Running from the UNIX Prompt

Enter the following command line:

```
csh -f $aleph_proc/p_ouf_load <parameter_list>
```

```
where <parameter_list> ::= <library>,<library_type>,<input_file>,<reject_file>,  
                           <log_file_ok>,<log_file_wrn>,<log_file_err>,  
                           <update_mode>,  
                           <fix_a>,<fix_u1>,<merge>,<fix_u2>,  
                           <update_db>,  
                           <bib_iln>,<aut_type>,<hol_related_bib>
```

54.1.2 Parameters Description

<library>

The Aleph library code (example: USM01). Mandatory.

<library_type>

Type of the library. Possible values: “BIB”, “AUT”, “HOL”. Mandatory.

<input_file>

Name of input file with doc records in Aleph sequential format. Mandatory.

<reject_file>

Name of output file for rejected records. Optional.

<log_file_ok>

Name of output file for logging information of records processed without errors or warnings. Mandatory.

<log_file_wrn>

Name of output file for logging information of records processed with warnings. Mandatory.

<log_file_err>

Name of output file for logging information of records processed with errors. Mandatory.

<update_mode>

Update action. Possible values: “A” (add), “U” (update), “D” (delete), “L” (linked). Mandatory.

<fix_a>

Name of fix routine for adding records. The fix routine has to be defined in tab_fix in \$data_tab of <library>. Optional.

<fix_u1>

Name of fix routine for updating records. The fix routine has to be defined in tab_fix in \$data_tab of <library>. This routine is called before a merge is done. Optional.

<merge>

Name of merge routine. The merge routine has to be defined in tab_merge in \$data_tab of <library>. Optional.

<fix_u2>

Name of fix routine for updating records. The fix routine has to be defined in tab_fix in \$data_tab of <library>. This routine is called after a merge is done. Optional.

<update_db>

A Y/N-flag indicating if the update in the database should be done. Possible values: “Y”, “N”. Mandatory.

<bib_ilm>

Field code of the Internal-Library-Number (ILN). Relevant only when library_type = "BIB". Optional.

<aut_type>

Type of the authority data. Relevant only when library_type = "AUT". This parameter is used only for logging information. Optional.

<hol_related_bib>

Library code of the related BIB library when library_type = "HOL". Mandatory if when library_type = "HOL".

Examples:

```
csh -f $aleph_proc/p_ouf_load      USM01,BIB,bibin.mrc,bibout.reject,biblog.ok,
biblog.wrn,biblog.err,A,Oufa,OUFU1,OUFM,OUFU2,Y,ilm,,
```

```
csh -f $aleph_proc/p_ouf_load      USM60,HOL,holin.mrc,,hollog.ok,
hollog.wrn,hollog.err,U,,,,Y,,,USM01,
```

55 Preventing the Automatic Creation of PAR Reciprocal Links Between Records

Two BIB records can be linked based on the LKR field cataloged in one record.

After you set tab_z103 and catalog LKR\$aPAR (parallel link types) in one BIB record, the system automatically creates a reciprocal PAR link in the corresponding linked record.

There is an option to set Aleph to prevent the automatic creation of reciprocal links for PAR type linkage. To configure this, set the update_z103_lkr_extended routine in the tab_z103 library table with the parameter lway-par.

For example: ./usm01/tab/tab_z103

```
!           1           2
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!->
update_z103_lkr_extended           lway-par
```

The above setup prevents the creation of the reciprocal link so that the Z103 link is not created for the corresponding record. The link will be one-way – from the current record to the corresponding record – but not vice-versa.

In order to make the link reciprocal, you must manually catalog a LKR in the linked record.

Note that modifying tab_z103 requires re-starting UTIL/E/1 (update doc index daemon); otherwise, the changes in tab_z103 do not take effect.

56 Generating a Locally Assigned Call Number In Bibliographic and Item Records

The fix routine, `fix_doc_090_call_no`, assigns a local call number to the 090 field of the BIB record using a running sequence counter.

56.1 Creating the Call Number in the BIB Record

The sequence is automatically assigned, depending on the prefix that the user inserts.

To activate the routine, set up `bib_lib/data_tab/tab_fix` as follows:

1	2	3
!!!!!!-	!!-	!!>
CALL	<code>fix_doc_call_no_090</code>	<code>MODE=PREVIEW</code>
INS2	<code>fix_doc_call_no_090</code>	<code>MODE=UPDATE</code>

In the above example, the CALL instance is activated manually. To activate it, you must first define it in `bib_lib/pc_tab/catalog/fix_doc.lng`

Column 3 of `tab_fix` contains the parameters of the fix. The MODE parameter is mandatory. It has two options:

- PREVIEW – After the sequence number a text is added in parenthesis. The sequence number is not yet stored in the database (Z311). This is to allow for a check of the new prefix before the database is updated.
- UPDATE – The sequence is stored in the database. If the sequence number has the added text, it is removed.

By default, the added text is “(expected)”. To change the text, use the parameter “TXT=” with the defined string as the text to be added. Note that if you change the text, you must add the TXT parameter to both modes – UPDATE and PREVIEW.

In order to activate this mechanism, the Z311 Oracle table must be created in the BIB library using `util/a/17/1`.

The following is an example workflow:

1. The user inserts the prefix TRN-T in subfield 090\$a.
2. The user activates `fix_doc_090_call_no`.
3. The system automatically assigns 090\$a with the next sequence – for example, TRN-T14(expected).
4. When the BIB record is saved, the appending text “(expected)” is removed and the 090\$a subfield is set with TRN-T14.

If the user enters a prefix ending with numbers (for example, TRN-T66), the system relates to 66 as the sequence number and does not assign a new sequence number.

Note that when activating this fix with a loader, the MODE parameter must be set to UPDATE only. In this case, the TXT parameter is not relevant.

It is possible to copy the call number from the 090\$a subfield of the BIB record to the call number type field of the Item record. For more information, refer to the

Copying the Call Number from the BIB Record to the Item Record section of the *Aleph 21 System Librarian's Guide - Items*.

56.2 Check Routine for Call Number Prefix

The check_doc_call_no_090 check routine reviews the prefix entered in subfield 090\$a and provides a record check warning when a new prefix is used. The message "New Prefix Alert" is displayed each time a new counter is used. Note that if you defined the TXT parameter in the fix routine, the same text must also be defined in the check.

The check routine is set in ./bib_library/tab/check_doc.

```
!          1          2          3
!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
CATALOG-INSERT          check_doc_call_no_090

Alert text is set in    ./aleph/error_eng/check_doc
! 1 2          3
!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
1100 L ($1$a): Call number prefix $2 is a new entry.
1101 L ($1$a): Call number prefix $2 is a new entry, with manual input of
the sequence=$3 !
1102 L ($1$a): Call number prefix $2 is a old entry, with manual input of
the sequence=$3 !
1105 L ($1$a): Error, Call number prefix $2 is an old entry, but maximum
for sequence reached; no automatic sequence handling !
1106 L ($1$a): Error, Call number prefix $2 to long (l'max=25); no
automatic sequence handling !
1107 L ($1$a): Error, Manual input of sequence $2 > 99999999; no automatic
sequence handling !
1110 L ($1$a): Error, unknown check status of the Call number $2, create an
issue !
```

57 Automatic Creation of 6XX Fields

This chapter describes the activity of the Create Additional Subject Heading(s) from Authority (manage-46) cataloging batch service.

The service creates a translated subject heading within the bibliographic record based on authority matching indexes. This can be used, for example, for updating bibliographic records with French RVM subject headings based on matching terms in the authority heading.

The Manage-46 service creates the 6XX bibliographic field by finding a match with the authority records and applying the 1XX/7XX authority field.

The service works in conjunction with the bibliographic library table tab_bib_aut_match that points to the relevant AUT database and index codes to which the match is performed.

57.1 The Batch Service: Create Additional Subject Heading(s) from Authority (manage-46).

The procedure that creates 6XX is submitted by the Create Additional Subject Heading(s) from Authority (manage-46) batch service. The service can be accessed via **Cataloging>Bibliographic Library Services> Catalog Maintenance Procedure**.

Batch parameters:

1. **Input File** – Mandatory. The name of the file that contains the bibliographic document numbers that are handled by the batch procedure. The input file must exist in the \$alephe_scratch directory. An input file can be generated by submitting various cataloging batch services, such as Retrieve Catalog Records (ret-01), Cross Files (ret-10), etc.

The following are examples of an input file (this is the format of the file produced by ret-<nn> services):

- o 000000130USM01
 - o 000000131USM01
 - o 000000132USM01
 - o 000000133USM01
 - o 000000134USM01
2. **Output File** – Mandatory. The name of the file in which you want to save the output file. The output file contains the bibliographic document numbers for which no translation occurred. The file can be found later in the \$alephe_scratch directory and be used for further processing (for example, print, load in cataloguing client, Search tab, etc).
 3. **Create 6XX using: AUT 1XX or AUT 7XX** – Mandatory. This selection determines whether the process will create the new subject heading 6XX using the AUT 1XX tag (can be used for English to French translation) or using the AUT 7XX tag (can be used for French to English translation).
 - o If 1XX is selected, the system applies the tab_bib_aut_match table in order to find the BIB 6XXx0/2/7 tags that should be translated and the

Authority library plus headings index code that is used for finding a matching authority record.

- If 7XX is selected the system applies the tab_bib_aut_match table in order to find the BIB 6XXx6 tags that should be translated and the Authority library code. The system always uses the GEN heading index of the AUT library for finding a matching authority record.
4. **Update Database** – Mandatory. You can decide whether updated records with new 6XX fields are updated immediately by selecting **Yes**) or whether you want to produce the report information in the standard log file only by selecting **No**). If you select **No**, then you can check the first run. If the run is ok, you can repeat the same run with Update Database= Yes.

57.2 Defining the AUT Index code for Detecting the AUT Headings - tab_bib_aut_match

The Aleph configuration table ./<bib_library>/tab/tab_bib_aut_match defines the Bibliographic tag that should be translated and sets the AUT index code to which the BIB tags are compared in order to find the authority record that can be used for translation.

The tab_bib_aut_match table is used by the Create Additional Subject Heading(s) from Authority (manage-46)service. The following is a partial example of ./usm01/tab/tab_bib_aut_match:

```
! TABLE_KEY 1,2
! COL 1. 5; ALPHA_NUM, UPPER; #;
!     BIB Tag & indicator;
! COL 2. 10; TEXT; ;
!     For second position 7 only: Subfield 2 content;
! COL 3. 5; ALPHA_NUM, UPPER; ;
!     AUT library code;
! COL 4. 5; ALPHA_NUM, UPPER; ;
!     AUT Index Heading code.Irrelevant for 6XX 2nd indicator 6;
! COL 5. 2; ALPHA_NUM, UPPER; ;
!     Filing procedure;
!     Filing procedure for the “text comparison” action.
!     Used for creating a temporary Z01-NORMALIZED-TEXT from
!     Z01-DISPLAY-TEXT without deletion of subfield codes.
!     Default is 99 of the AUT library (make sure it doesn’t
!     include filing_del_subfield nor filing_del_subfield_code
!     routines);
! 1 2 3 4 5
!!!!-!!!!!!!!-!!!!-!!!!-!!
650#0      USM10 LCS
650#2      USM10 MLC
650#7 aat  USM10 AAT
6###6     USM10
```

Explanation of tab_bib_aut_match activity

1. The content of the Bibliographic record's tag+indicator (set in col. 1) is compared with the Authority heading set in the Authority index code (col.4).
2. Col. 2 is relevant only when second indicator is 7 (e.g. col.1 is set with 650#7). If col.2 is populated, the system ensures that subfield 2 of the tag+indicator (set in col. 1) contains exactly the text entered in col.2. If the bibliographic records subfield 6XXx7 is not identical to col. 2, no translation occurs.
3. When activating manage-46 using "Create 6XX using AUT 1XX" (English to French), the system looks for the Authority index (col.4) in the Authority library (col. 3).
4. When activating manage-46 using "Create 6XX using AUT 7XX" (French to English), the system looks for the Authority library (col.3) in which GEN index is used. Therefore, col. 4 (Authority index) is irrelevant (GEN index is always used).
5. When activating manage-46 using Create 6XX using AUT 7XX (French to English), the system looks for the Authority library (col.3) and uses GEN Authority index (hard-code). Therefore, col. 4 (Authority index) is irrelevant.
6. The Authority index code (col.4) must be defined and properly set in the Authority database (tab11_acc, tab00.Ing, etc).
7. For matching the BIB 6XX to the Authority index, the system activates the filing routine set in col. 5 of tab_bib_aut_match. The filing routine should be set in tab_filing of the AUT library. This routine should not include del_subfield/del_subfield_code so that text is normalized without deletion of subfield codes. The default filing routine is 99. This way, the system compares the text of the bibliographic record (tag+subfield+indicator) and the authority indexed heading in order to determine if a match is detected.
8. The library should set the order of the lines in tab_bib_aut_match according to the translation priority preferences so that, for example, if the bibliographic record does not contain 650x0, the system continues to look for 650x2. If it does not exist, the system continues to look for 650x7 with subfield 2=aat.
9. The library may set multiple lines per bibliographic tag+indicator. The system should prefer 6XXx0, 6XXx2 or 6XXx7 (in that order) and stop the process when a translation for one type of code has been found. However, the process attempts to find translations for all of the occurrences of this preferred tag.

Note: Since the AUT indexes for creating the French 6XX is built through regular indexing setup and processes, with a separate index for 750x0, 750x2 and 750x7 \$\$2=aat. The filing procedure for these indexes should NOT include del_subfield nor del_subfield_code filing routines and therefore the subfield code will be retained in the AUT Z01 lists that are built for this purpose.

57.3 Manage-46 Service Functionality - Workflow and Example

1. The library prepares in advance an input file using standard Aleph retrieval options, for example, records added from date-to-date or records from system number-to-system number, etc. Such input files can be automatically generated by Aleph standard cataloging batch services. The manage-46 batch service processes each record contained in the input file records.
2. The user activates the service for AUT 1XX or AUT 7XX translation (user selection within service interface).
 - If AUT 1XX is selected, the system applies `tab_bib_aut_match` to find the Bibliographic fields that should be translated and the authority and index codes to which to make the text comparison. The bibliographic text that should be translated is normalized using the filing routine set in `tab_bib_aut_match` (filing routine that do not delete subfield codes).
 - If AUT 7XX is selected, the system applies `tab_bib_aut_match` to find the Authority library (e.g. USM10) in which GEN index code is used for text comparisons. The bibliographic record tag 6XXx6 that should be translated is normalized using the filing routine set in `tab_bib_aut_match` (a filing routine that does not delete subfield codes).
3. The Bibliographic normalized text is compared with the AUT index text (as defined in `tab_bib_aut_match` or GEN index). The comparison is made according to the match algorithm (see details below).
4. If a single authority record is found, the BIB is enriched with a new 6XX.
 - If AUT 1XX is selected - a new 6XXx6 is created
 - If AUT 7XX is selected - a new 6XXx0 is created

The newly created field is written directly after the source field.

5. The procedure is repeated for each of the 6XX fields. If no translation whatsoever occurs, the BIB record number is reported in the batch output file. The output file is in standard Aleph retrieval format and can be used for further processing. The following is an example of an output file:

```
000000131USM01
000000132USM01
```

57.4 Match Algorithm and Translate

Following is a description of the match and translate algorithm for both translations:

- Create 6XX using 1XX (English to French)
- Create 6XX using 7XX (French to English)

57.4.1 Match and Translate for Create 6XX Using 1XX

When a user activates the service with “Create 6XX - using 1XX” (English to French):

1. The batch process of a record first checks if there is at least one 6XXx6 field in the BIB record (Laval's French subject heading):
 - If yes, (i.e. there is a 6XXx6), the record is not processed. This record is not listed in the output file.
 - If no, (i.e. there is no 6XXx6), the system activates the "match algorithm" in order to create a new 6XXx6 field.
2. The fields listed in columns 1 and 2 of tab_bib_aut_match are the candidate fields for translation. The library should set the lines in tab_bib_aut_match in the following order so that the translation occurs for:
 - 6XXx0, or
 - 6XXx2 if there is no 6XXx0, or
 - 6XXx7 containing \$\$2 aat, if there is no 6XXx0 or 6XXx2
3. Text normalization (filing routine) is activated for the fields.
4. Search full text string for match: BIB 6XX is compared with the AUT index.
 - If the procedure runs smoothly (the entire text is translated and there are no multiple possible translations), a new 6XXx6 field is created using the 1XX of the matching AUT record. The record is not listed in the output file. The newly created 6XXx6 field is written directly after the source field.
 - If a translation is not found for the first subfield, the procedure stops for this field. The system checks other fields. If no other fields are translated (a new 6XXx6 is not created), the record is listed in the output file.
 - If multiple translation records are found, the procedure stops for this field. If no other fields are translated (new 6XXx6 is not created); the record is listed in the output file.

When checking for a match:

- If a match is not found, the system drops the last subfield and repeats the search. The dropped subfield and its position are kept in a buffer for a separate search.
- If a match is not found, the system drops the last two subfields and repeats the search. The subfield buffer is cleared and dropped subfields and their position are kept in a buffer for a separate search.
- If a match is found, and the subfield buffer is not empty, the system deals with the subfield buffer in same manner.
- If there is a non-translated subfield, the system cuts off the field from that subfield to the end. This is still considered a successful match.
- If the first subfield is not translated, or if there are multiple options for any of the strings, this is considered a non-successful match.
- If the process does not create at least one successful match for a single record, the document number is written on the output file.

In this process, subfield codes should match and the geographic subdivision is treated the same as other subdivisions. The system keeps the order of subfields as in the original text.

Example cases:

Case 1: - Full “match” is found and BIB 650x6 is created with AUT 1XX content.

BIB record 000000123 contains an English Subject heading:

650_0 \$a Gardens
 \$x Styles
 \$x History
 \$y 20th century
 \$v Pictorial works

The Authority database contains a matching French heading (7XX_0) derived from 3 different records:

AUT record 000000001

150__\$a Jardins
 \$x Styles
750_0 \$a Gardens
 \$x Styles

AUT record 000000002

180_0\$x Histoire
 \$y 20e siècle
780_0\$x History
 \$y 20th century

AUT record 000000003

185_0\$v Ouvrages illustrés
785_0\$v Pictorial works

manage-46 activation causes an update of the BIB record with a new field. The source field is retained as is:

650_0 \$a Gardens
 \$x Styles
 \$x History
 \$y 20th century
 \$v Pictorial works
650_6 \$a Jardins
 \$x Styles
 \$x Histoire
 \$y 20e siècle
 \$v Ouvrages illustrés

Case 2 – Partial “match” is found.

BIB record 000000123 contains an English Subject heading:

650_0 \$a Gardens
 \$x Styles
 \$x History
 \$y 20th century

\$v Pictorial works

The Authority database contains:

AUT record 000000001

150__ \$a Jardins

\$x Styles

750_0 \$a Gardens

\$x Styles

AUT record 000000003

185_0\$v Ouvrages illustrés

785_0\$v Pictorial works

AUT record 000000004

185 0 \$x Histoire

785 0 \$x History

The Authority database does NOT contain a record for “\$x History \$y 20th century” nor “\$y 20th century”

manage-46 activation causes an update of the BIB record with a new field. Source field retains as is:

650_0 \$a Gardens

\$x Styles

\$x History

\$y 20th century

\$v Pictorial works

650_6 \$a Jardins

\$x Styles

\$x Histoire

57.4.2 Match and Translate for Create 6XX Using 7XX

When user activates the service with “Create 6XX - using 7XX” (French to English):

1. The batch process of a record first checks if BIB record contains 6XXx0 or 6XXx2 or 6XXx7 + \$\$2=at
 - o If yes, record is not processed. This record is not listed in the output file.
 - o If no, the system activates the “match algorithm” in order to create new 6XXx0 field.
2. The BIB record 6XXx6 field is used to match with the GEN Authority index of the Authority library that is set in column 3 of tab_bib_aut_match.
3. Text normalization (filing routine) is activated for the BIB 6XXx6 field (set in col. 5 of tab_bib_aut_match (make sure it does not include del_subfield/del_subfield_code routines).

4. The service searches full text string for a match. BIB 6XX is compared with the GEN index.
5. If the procedure runs smoothly (the entire text or part of it is translated and there are no multiple possible translations), a new 6XXx0 field is created. The content of the field is based on the AUT 7XXx0 (LC subject heading) of the matching Authority record. The record is not listed in the output file. The newly created 6XXx0 field is written directly after the source field.
6. If multiple translation records are found, the procedure stops for this field. If no other fields are translated (new 6XXx0 is not created), the record is listed in the output file.

When checking for a match:

- The entire string is searched for and if it is not found, the last subfield code is cut off, cursively. The search stops when a match is found.
- No buffer for remaining non-matched is required (i.e. no need to keep non-translated subfields for a separated translation).
- Non-found subfields are dropped.

In this process, subfield codes should match and geographic subdivisions are treated the same as other subdivisions, i.e. if they are not in RVM (AUT), subfields are cut-off. The system should keep the order of subfields as in the original text.

Example case:

“Match” is found for a single subfield and BIB 650_0 is created with AUT 7XX content.

BIB record 000000123 contains French Subject headings:

```
650_6 $a Jardins
      $x Styles
      $x Historie
      $y 20e siècle
      $z Ouvrages pictoriales
```

The authority database contains a matching English heading (1XX_0) derived from 1 record:

```
AUT record 000000001
150__ $a Jardins
750_0 $a Gardens
```

manage-46 activation causes an update of the BIB record with a new 650_0 field. The source field remains as is.

```
650_6 $a Jardins
      $x Styles
      $x Historie
      $y 20e siècle
      $z Ouvrages pictoriales
```

58 Automatic Translation of Bibliographic Note Fields

Aleph can automatically translate bibliographic note fields from one language to another. To enable this feature, configure the following files:

- Fix record routine: `fix_doc_notes` (set in `./xxx01/tab/tab_fix`)
- Aleph table: `tab_fix_notes` - list of translations, per Bibliographic tag and subfield (`./xxx01/tab/tab_fix_notes`).

58.1 Fix Routine for Translation - `fix_doc_notes`

To configure the automatic translation of Bibliographic fields, set `fix_doc_notes` routine in `tab_fix`.

The following is an example of `./usm01/tab/tab_fix`

```
! 1                2                3
!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
INS2  fix_doc_notes
```

As with all other fix routines, you can set the fix routine (in column 1 of `tab_fix`) to be invoked in various routines, for example, online record handling, batch loaders, and batch updates) so that its activation is part of the bibliographic record management.

This fix routine replaces the text within a bibliographic field with a translated text defined in the `tab_fix_notes` Aleph configuration table.

58.2 Setting Up a List of Translations - `tab_fix_notes`

The Aleph configuration table, `./xxx01/tab/tab_fix_notes`, contains a list of translations per tag and subfield. The library may edit this table to include translations of relevant tags and subfields.

The following is a partial example of `./usm01/tab/tab_fix_notes`. (The actual lengths of columns 3 and 4 are longer than that what they appear in the following table structure.)

```
! TABLE_KEY 1,2,3
! COL  1.   5; ALPHA_NUM, UPPER; #;
!           Tag & indicator;
! COL  2.   1; ALPHA_NUM, LOWER;  ;
!           Subfield;
! COL  3. 150; TEXT;  ;
!           Source text;
! COL  4. 150; TEXT;  ;
```

Translated Text;				
!	1	2	3	4
!!!!-!-!!!!!!>-!!!!!!>				
245##	h	[electronic resource]		[ressource électronique]
310##	a	Annual		Annuel
310##	a	Biannual		Semestriel
310##	a	Bi-annual		Semestriel
310##	a	Biennial		1 no par 2 ans
310##	a	Bimonthly		Bimestrie
500##	a	Adaptation of :		Adaptation de :
500##	a	Added t.p. title addit. :		Titre de la p. de t.
500##	a	At head of title		En tête du titre
500##	a	Caption title		Titre de départ
500##	a	Cataloguer's title		Titre du catalogueur
500##	a	Colophon title d'imprimer		Titre de l'achevé
500##	a	Cover title		Titre de la couv.

58.3 Automatic Translations – Functionality and Examples

The following is a description of the automatic translate feature that is implemented when fix_doc_notes is activated for a bibliographic record:

58.3.1 Compare Action

The system compares the source table tab_fix_notes values with the bibliographic record. The fix routine (fix_doc_notes) reads the source list (tab_fix_notes) and compares it with the bibliographic record relevant tag + subfield left-anchored text.

Notes:

- The source file has to list the lines for a specific tag one after the other. For example, if there are 10 translation lines for tag 500\$a, group all lines (similar to the example above of tab_fix_notes).
- In the source file, list longer phrases before shorter ones.
- The comparison is not case sensitive.

58.3.2 Replace Action

1. The matched text is replaced with the translated text as defined in the source file.

Notes:

- The phrase in English must exist in its entirety in the bibliographic record in order for the translation text to replace it.
- Text that is not left aligned is not translated.

2. Parts of a field that do not have a matching string in the source file are copied as is into the translated field.
3. The source field is overwritten.

For example:

The following is an example of the `tab_fix_notes` source:

```
! 1      2                      3                      4
!!!!-!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!->
500## a Title from container                Titre du conteneur
500## a Title romanized:                    Titre converti en
écriture latine:
500## a Title                               Titre
```

Case 1:

BIB record 000000123 contains the following text:

```
500## $$aTitle from container
```

`tab_fix_notes` contains the following text:

```
500## a Title from container                Titre du conteneur
```

This causes an automatic translation of the note to:

```
500## $$aTitre du conteneur
```

Case 2:

BIB record 000000456 contains the following text:

```
500## $$aTitle from back cover
```

There is no translation for `Title from back cover` so the following line in the source field:

```
500## a Title                               Titre
```

changes the note field to:

```
500## $$aTitre from back cover
```

Case 3:

Bibliographic record 000000789 contains the following text:

```
500## $$aAdditional Title from container
```

There is no translation for the left aligned text `Additional`; therefore, the line does not change:

```
500## $$aAdditional Title from container
```


<i>Enum. BibUnit</i>	<u>863</u>	<u>40</u>	<u>8</u>	.1
			a	4
			b	3-88
			i	2013
			j	01-03
			k	03-29
			w	g
			9	Y
<i>Enum. BibUnit</i>	<u>863</u>	<u>40</u>	<u>8</u>	.2
			a	4
			b	95-97
			i	2013
			j	04
			k	05-07
			w	g
			9	Y
<i>Txt.Hold BibUnit</i>	866	<u>41</u>	<u>a</u>	v.4:no.3-88(2013:Jan.03-Mar.29),
			<u>8</u>	.1
<i>Txt.Hold BibUnit</i>	<u>866</u>	<u>41</u>	<u>a</u>	v.4:no.95-97(2013:Apr. 05-07),
			<u>8</u>	.2

There is an option to set the above UPDATE-ENUM section with message type (column 2): o

!	1	2	3	4	5	6	7	8	9	10	11
12											
!!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-!!!!!!-											
!!!!!!-!!!!!!											
UPDATE-ENUM		o	USM60								

Message type o: update HOL record with 86X info (Summary Holdings) using `expand_doc_hol_86x_iso` expand routine. (For more information about `expand_doc_hol_86x_iso`, see the *Aleph 22 System Librarian Guide-Indexing*).