ALEPH VERSION 19.01 and Later

How to Set Up an ALEPH Union Catalog

The bridge to knowledge
Last Update: May 6, 2008
Document Version 1.2
Code: A-ver19-HTUC-1.2

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.

Any references in this document to non-Ex Libris Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Ex Libris product and Ex Libris has no liability for materials on those Web sites.

Documentation produced January 2008
Document version 1.2
Web address: http://www.exlibrisgroup.com

Table of Contents

1 Introduction

1.1 What is a Union Catalog?

A Union Catalog is a library catalog that contains the holdings of more than one library. The Union Catalog often serves as a utility for resource discovery, allowing users to search the catalogs of multiple institutions. Frequently, the Union Catalog offers some form of inter-library loan so that users can request the resources they identify.

1.2 How is a Union Catalog different from a Standard Catalog?

In ALEPH, a Union Catalog differs from a standard library catalog in a number of significant ways. A Union Catalog contains the holdings of multiple libraries and thus contains multiple records for titles that are held by multiple institutions. A Union Catalog does not contain item records. A Union Catalog has embedded holdings instead of linked holdings records. Finally, Union Catalogs are characterized by frequent loads of records from contributing libraries.

1.2.1 Duplicate Records

ALEPH uses a sophisticated algorithm that it developed in conjunction with California Digital Library to identify duplicate records. In the OPAC, result sets are de-duplicated using pre-constructed record equivalency tables and records are merged on the fly for display.

1.2.2 Embedded Holdings

Bibliographic records in the Union Catalog contain embedded holdings statements. These embedded holdings generate a location display that includes Campus/Location, Library/Sub-location, and call number. In addition, the Union Catalog offers link to functionality for displaying up-to-date circulation status information from the local library. Circulation status can be obtained using Z39.50 or the Ex Libris X-server.

1.2.3 Ongoing Loads

Libraries that contribute records to a Union Catalog need to constantly update their libraries holdings in the Union Catalog. The result is that they send frequent batches of new and updated records. These batches need to be processed efficiently. ALEPH has a set of batch processes that convert the incoming records and load them into the Union Catalog.

1.3 This document

This document describes in detail how to set up the various components of an ALEPH Union Catalog, including the special features mentioned above. The document focuses on configuration and functionality that is available from version 16.02 and on. Please see earlier versions of the document to configure earlier versions of the product.

2 Equivalent Records

2.1 What are equivalent records?

Equivalent records are bibliographic records that describe the same bibliographic entity. Typically, equivalent records have been contributed by different Union Catalog members and thus have identical or very similar bibliographic data but very different holdings data. [In the Union Catalog, holdings data is embedded in the bibliographic record]. ALEPH uses an algorithm that was developed in conjunction with Melvyl (California Digital Libraries), to identify equivalent records. Equivalent records are not physically merged; instead, lists of pre-built equivalencies are stored in Oracle tables and these tables are used to de-duplicate retrieval sets and merge equivalent records on the fly.

2.2 Storing equivalent records

There are two Oracle tables that are used for storing equivalent records, the Z120 record and the Z127 record. Each record in the catalog has its own Z120 and Z127 record. For each record in the database, the Z120 record stores a complete list of records that are equivalent to it, as well as a single record that is considered the preferred record. Batch processes create the record equivalency table and maintain it so that whenever a bibliographic record is updated its equivalencies are rebuilt. Both the list of records that it is considered equivalent to and the list or records that contain it as an equivalent record are updated.

The Z120 table contains the following data elements: system number, total number of equivalent records, system numbers of all equivalent records, system number of preferred record, and update flag. Z120 records are built after the initial data load (using p_union_01 and p_union_02) and then updated each time the record is updated. The process that updates Z120 records is called p_union_04.

Illustration 1 – Z120 table definition for version 14.2 –15.5

```
          02 Z120-REC-KEY.
            03 Z120-DOC-NUMBER                PICTURE 9(9).
          02 Z120-REC-KEY-1.
            03 Z120-PREFERRED-DOC-NUMBER      PICTURE 9(9).
          02 Z120-UPDATE-FLAG                 PICTURE X(1).
          02 Z120-SAME-NO-LINES               PICTURE 9(3).
          02 Z120-SAME OCCURS 100.
            03 Z120-SAME-DOC-NUMBER           PICTURE 9(9)
```

Illustration 2 – Z120 table definitions for version 16.02 and later

```
        01  Z120.
          02 Z120-REC-KEY.
            03 Z120-DOC-NUMBER                PICTURE 9(9).
          02 Z120-REC-KEY-1.
            03 Z120-PREFERRED-DOC-NUMBER      PICTURE 9(9).
          02 Z120-UPDATE-FLAG                 PICTURE X(1).
          02 Z120-SAME-NO-LINES               PICTURE 9(3).
          02 Z120-SAME OCCURS 500.
            03 Z120-SAME-DOC-NUMBER           PICTURE 9(9).
```

The Z127 record, on the other hand, stores only the system number of a single record and its preferred record.

2.3 Equivalency Algorithm

The equivalency building algorithm goes through three phases in the process of determining equivalent records and building the equivalency table (the Z120 table). They are:

- Candidate Selection
- Record Matching
- Preferred Record Selection

Each of these phases is controlled by a combination of programs and configuration tables. There is one central configuration table that controls which combination of programs and tables are used. This table is located in alephe_tab and is called union_global_param.

Illustration 3: Union_global_param [alephe_tab]

```
USM90 B candidate_prog        union_candidate_cdl
                       100
USM90 B match_prog            union_match_cdl
USM90 B preferred_prog        union_preferred_cdl
USM90 B normalize_prog        union_normalize_cdl
```

2.3.1 Phase One - Candidate selection

During the Candidate Selection phase of the equivalency building process, potential matching records are retrieved. Two limits are involved in the process. One is a configurable limit that determines whether a refinement on the original search is done. If more than a given number of records are retrieved, the set is refined by adding the year of publication or place of publication to the search.
This limit is defined in $alephe_tab/union_global_param in the sixth column of the candidate program line.
Illustration 4 – `union_global_param [alephe_tab]`

```
USM90 B candidate_prog        union_candidate_cdl
                       100
```

The second limit defines the maximum number of candidates after the refinement process. This is a fixed limit. In version 14.2 through 15.5 it cannot be larger than 100; in version 16.02 and later, it can be up to 500. If there are more candidates than this limit, the record is considered as having no match, and the check for a match is not performed.

The candidate phase of the equivalency process utilizes three direct indexes - LCCN, ISSN, and ISBN and three keyword indexes – NTL (Normalized Title), WYR (year),

and WPL (place of publication) More information about set up of these indexes can be found in section 2.3.4 Direct indexes and 2.3.5 Keyword indexes.

The following indexed fields are used to retrieve candidate records:

- LCCN (010 $$ a and $$z)
- ISSN (022 $$a $$y and $$z) or ISBN (020 $$a and $$z)
- Normalized Main title (245 $$a, $$b, $$n, and $$p)
- Year of Publication (008 position 07-10) or Place of Publication (260 $$a)

Note that there is an "or" relationship between the first three search terms (LCCN **or** ISSN/ISBN **or** Title), and there is an "and" relationship between them and the fourth term ((LCCN **or** ISSN/ISBN **or** Title) **and** Year of publication/Place of publication).

The fourth term is used only when the candidates number is more than the number in the sixth column of candidate_prog in $alephe_tab/union_global_param.

Year of publication from field 008 positions 07-10 is used to refine the candidate search for non-serial records. It uses the WYR index.

Place of publication is used to refine the candidate search for serial records. Place of publication data undergoes normalization according to filing code 75. It uses the WPL index.

It is also possible to choose another index code for candidate refinement. The codes used for performing a refined search can be written in the fifth column of $alephe_tab/union_global_param. First the code for non-serial records is written, then a comma, and then the code for serial records. If either of the index codes is left blank or if the entire column is empty, the default index code is taken (i.e. WYR and WPL).

Illustration 5 – `union_global_param [alephe_tab]`

```
USM90 B candidate_prog       union_candidate_cdl
WAU, WTI                                  100
```

2.3.2 Server tables

The union_global_param table controls the individual components of the equivalency algorithm and is located in the alephe_tab directory with other shared tables. In the table, programs and parameters are set for each of the phases of the equivalency building process. Currently, there is only one program that can be called for the candidate retrieval phase of the equivalency building program. It is union_candidate_cdl. Thus, the following line should be entered in `union_global_param` for your Union Catalog library.

Illustration 6 – `union_global_param [alephe_tab]`

```
USM90 B candidate_prog       union_candidate_cdl
                      100
```

The sixth parameter, number of candidates, is a three digit number. It determines whether a refinement on the original search is made (see section 2.3.1 Phase One - Candidate selection).

2.3.3 Library Tables

The candidate retrieval phase of the record equivalency algorithm uses library indexes to retrieve potential duplicates. Thus, the Union Catalog library must have the direct and keyword indexes set up in its indexing tables.

2.3.4 Direct indexes

The ISSN, ISBN, and LCCN fields need to be indexed as direct indexes. They should go through expand_doc_extract so that each subfield is indexed separately. They must also be normalized so that prefixes and suffixes are deleted and hyphens are removed. It is especially important that parentheses be removed by the filing routine. Failure to remove parentheses can result in errors during the p_union_02 and p_union_04 processes.

Here are direct index specific settings that should be found in all Union Catalog libraries.

Illustration 7 – `tab11_ind [data_tab]`

```
a010                    010
z010                    010
a020                    020
z020                    020
a022                    022
y022                    022
z022                    022
```

Illustration 8 – `tab_expand [data_tab]`

```
INDEX      expand doc extract
```

[data_tab]

Illustration 9 – `tab_expand_extract [data_tab]`

```
010## a a010
010## z z010
020## a a020
020## z z020
022## a a022
022## y y022
022## z z022
```

Illustration 10 – tab00.eng [data_tab]

```
H 010   IND     70 00       00       LCCN
H 020   IND     71 00       00       ISBN
H 022   IND     72 00       00       ISSN
```

Illustration 11 – tab_filing

```
!* 70 - LCCN normalization for LCCN index to be used by
!* Equivalency building program and union_match_cdl_mo
70   del_subfield
70   compress             -
70   compress_blank
70   non_numeric
!* 71 - ISBN normalization for ISBN index to be used by
!* Equivalency building program and union_match_cdl_mo
71   isbn
!* 72 - ISSN normalization for ISNN index to be used by
!* Equivalency building program and union_match_cdl_se
72   issn
```

2.3.5 Keyword indexes

The main title (245) must be sent to a keyword index called NTL. This index is used for title searching in the Candidate phase of equivalency building. Before indexing a title, an expand program called `expand_doc_ntl` normalizes the title by stripping initial articles (using non-filing indicators), stripping punctuation and diacritics, and compressing spaces. Finally, the first twenty-five characters are indexed as a single keyword. The expand_doc_ntl program uses filing routine 75 to normalize data in the bibliographic record. Be sure that this routine is defined in your tab_filing.

Illustration 12 – tab11_word [data_tab]

```
NTL##                                    03     NTL
```

Illustration 13 – tab_expand [data_tab]

```
WORD       expand_doc_ntl
```

Illustration 14 – tab00.eng

```
H NTL   W-035      00       06       Normalized Title
```

Illustration 15 – tab_filing

```
!* Reserved for equivalency building program - used to
create the normalized title (NTL)
74   del_subfield
74   non_filing
74   to_blank             !@#$%^&*()_+-={}[]:";<>?,./~`
74   char_conv            FILING-KEY-01
74   compress_blank
74   first_25
74   to_lower
```

In addition to the NTL index, libraries must have the word from publisher and year of publication indexes defined. The former is called WPL and the latter WYR.

2.4 Phase Two – Record Matching

The second phase in the equivalency building process is the record matching phase. During this phase, all candidate records are compared and those found to be equivalent are recorded in the Z120 record. While much of this process is hard-coded, there are a number of configuration tables involved and some configuration. This phase focuses on the CDL matching algorithm which is the basis for other matching algorithms.

2.4.1 Server tables

The following line should be entered in the server table `union_global_param`. This table controls the individual components of the equivalency algorithm.

Illustration 16 – `union_global_param [alephe_tab]`

```
USM90 B match_prog          union_match_cdl
```

2.4.2 Library tables

The program `union_match_cdl` uses three configuration tables; all three must be located in the data_tab directory of your Union Catalog library:

- tab_cdl_mo_weights
- tab_cdl_se_weights
- tab_com_tit_cdl

Two of these tables (tab_cdl_xx_weights) set the record equivalency threshold as well as points that are assigned for each phase of the record matching process. These are set separately for serial and non-serial records. The third table lists common serial titles for the library. Common titles receive fewer points for title matches than other titles.

Note that the equivalency algorithm permits there to be matches between all formats except Serials. Serial records can only match other Serial records; however, records with the format book can match records with the format music, computer file, and so on.

2.4.3 Assigning weights for record matching of monograph records (non-serial)

Each step in the record matching phase of the equivalency building algorithm compares data in the base record with data in every potential candidate.

The equivalency building algorithm of monographs is based on the following MARC fields: 008,010, 020, 245, 260, 300, 100, 110 and 111.

2.4.3.1 Date Section: Field 008 position 07-10

Positions 07-10 of field 008 contain a date.
The data is taken as is without any normalization.

1. If the last two positions of the date, positions 008/09-10, contain one of the values: "uu", "^^" or spaces (i.e. there is no exact date) in either record, then no points are given and the program exits the date section.

2. If the dates are an exact match then the points for "date exact match" in the tab_cdl_mo_weights table is added to the record equivalency sum and the program exits the date section.

Illustration 17 – `tab_cdl_mo_weights [data_tab]`

```
date exact match                                    + 200
```

3. If there is a two year difference between the dates (upwards or downwards) then the points for "date within 2" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the date section.

Illustration 18 – `tab_cdl_mo_weights [data_tab]`

```
date within 2                                       - 025
```

4. If the dates do not match, then the points for "date mismatch" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the date section.

Illustration 19 – `tab_cdl_mo_weights [data_tab]`

```
date mismatch                                       - 250
```

2.4.3.2 LCCN Section: Field 010 subfields a,z

Field 010 holds the LC control Number.
The data undergoes normalization according to filing code 70.

Illustration 20 – `tab_filing [data_tab]`

```
70   del_subfield
70   compress              -
70   compress_blank
70   non numeric
```

Subfields ‘a’ and ‘z’ are taken into account in this match. Only the first 5 instances of subfield ‘a’ or subfield ‘z’ are compared until they fit one of the steps listed below.

1. If there is no 010 field in one or both records then no points are given and the program exits the LCCN section.

2. If there is a match between subfield a and subfield a then the points for "010aa match" in the tab_cdl_mo_weights table are taken and the program exits the LCCN section.

Illustration 21 – `tab_cdl_mo_weights [data_tab]`

```
010aa match                                          + 200
```

3. If there is a match between subfield a and subfield z then the points for "010az match" in the tab_cdl_mo_weights table are taken and the program continues to the next match in this section.

Illustration 22 – `tab_cdl_mo_weights [data_tab]`

```
010az match                                          + 100
```

4. If there is a match between subfield z and subfield z then the points for "010zz match" in the tab_cdl_mo_weights table are taken.

Illustration 23 – `tab_cdl_mo_weights [data_tab]`

```
010zz match                                          + 100
```

5. The highest number between the points given in step 3 or in step 4 are taken and the program exits the LCCN section.

If no point is given in these steps, continue to the next step in this section.

6. If subfield a and subfield a do not match then the points for "010aa mismatch" in the tab_cdl_mo_weights table are taken and the program continues to the next match in this section.

Illustration 24 – `tab_cdl_mo_weights [data_tab]`

```
010aa mismatch                                       - 470
```

7. If subfield a and subfield z do not match then the points for "010az mismatch" in the tab_cdl_mo_weights table are taken.

Illustration 25 – `tab_cdl_mo_weights [data_tab]`

```
010az mismatch                                       - 050
```

8. The lower number between the points given in step 5 and step 6 is taken and the program exits the LCCN section.

2.4.3.3 ISBN Section: Field 020 subfields a,z

Field 020 holds the monograph's ISBN.
The data undergoes normalization according to filing code 71.

Illustration 26 – `tab_filing [data_tab]`

```
71   isbn
```

Subfields ‘a’ and ‘z’ are taken into account in this match. Only the first 5 instances of subfield ‘a’ or subfield ‘z’ from all 020 fields (not for every 020 field) are compared until they fit to one of the steps listed below.

1. If there is no 020 field in one or both records then no points are given and the program exits the ISBN section.

2. If there is a match between subfield a and subfield a then the points for "020aa" in the tab_cdl_mo_weights table are taken and the program exits the ISBN section.

Illustration 27 – `tab_cdl_mo_weights [data_tab]`

```
020aa                                                    + 085
```

3. If there is a match between subfield a and subfield z then the points for "020az" in the tab_cdl_mo_weights table are taken and the program continues to the next match in this section.

Illustration 28 – `tab_cdl_mo_weights [data_tab]`

```
020az                                                    + 030
```

4. If there is a match between subfield z and subfield z then the points for "020zz match" in the tab_cdl_mo_weights table are taken.

Illustration 29– `tab_cdl_mo_weights [data_tab]`

```
020zz                                                    + 010
```

The higher number between the points given in step 3 and in step 4 is taken and the program exits the ISBN section.

5. In all other cases, the points for "020mismatch" in the tab_cdl_mo_weights table are taken and the program exits the ISBN section.

Illustration 30 – `tab_cdl_mo_weights [data_tab]`

```
020mismatch                                              - 225
```

2.4.3.4 LCCN and ISBN Weights Comparison

The points from both LCCN and ISBN sections are not added to the record equivalency sum.

The points of one of them are added to the record equivalency sum according to the following algorithm:

1. If either one of the weights is zero, the non zero value is added.
2. When one is positive and the other negative, the sum of both values is added.
3. When both are positive, the highest value is added.
4. When both are negative, the lowest value is added.

2.4.3.5 Short Title Section: Field 245 subfields a,b,n,p

Field 245 contains the monograph's title.
The data undergoes normalization according to filing code 75.

Illustration 31 – `tab_filing [data_tab]`

```
75   del_subfield
75   to_upper
75   suppress
75   numbers
75   compress              '
75   to_blank              !@#$%^&*()_+-={}[]:";<>?,./~`
75   expand_num
75   non_filing
75   pack_spaces
75   char_conv             FILING-KEY-01
```

1. If the titles are an exact match then the points for "short title match" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the short title section.

Illustration 32 – `tab_cdl_mo_weights [data_tab]`

```
short title match                                    + 450
```

2. If the titles do not match, no points are given to the record equivalency sum and the program exits the short title section.

2.4.3.6 Evaluation Section

The points up until now are calculated into a value. This value is compared to the "treshold" line in the tab_cdl_mo_weights table.

Illustration 33 – `tab_cdl_mo_weights [data_tab]`

```
treshold                                             + 875
```

1. If the value is higher than the threshold, the records are considered as a match and the match process is stopped.

2. If the value is negative, the records are considered as a mismatch and the process is stopped.

3. If the value is positive and below the threshold, the process continues to the next step in the algorithm.

2.4.3.7 Full Title Section: Field 245 subfields a,b,n,p

Field 245 contains the monograph's title.
The data undergoes normalization according to filing code 75.

Illustration 34 – `tab_filing [data_tab]`

```
75   del_subfield
75   to_upper
75   suppress
75   numbers
75   compress            '
75   to_blank            !@#$%^&*()_+-={}[]:";<>?,./~`
75   expand_num
75   non_filing
75   pack_spaces
75   char_conv           FILING-KEY-01
```

1. If there was a match on the short title (see section 2.4.3.5 step1 "Short title match"), the points given for that match are subtracted.
2. If the titles are an exact match then the points for "full title match" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exists the full title section.

Illustration 35 – `tab_cdl_mo_weights [data_tab]`

```
full title match                                    + 600
```

3. If any or both normalized titles are shorter than 9 characters, no points are given to the record equivalency sum and the program exits the full title section.
4. If any of the titles is contained in the other title, the points for "full title occur within" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the full title section.

Illustration 36 – `tab_cdl_mo_weights [data_tab]`

```
full title occur within                             + 350
```

5. If there are identical words in both titles, and the percentage of these words is 50% or more of the longest title's words, and their order is not the same, then the following point calculation is made:
   ("full title keywords" * percentage of matching words)
   The points are added to the record equivalency sum and the program exits the full title section.

Illustration 37 – `tab_cdl_mo_weights [data_tab]`

```
full title keywords                                 + 450
```

6. If there are identical words in both titles, and the percentage of these words is 50% or more of the longest title's words, and their order is the same, then the following point calculation is made:
   ("full title keywords" * percentage of matching words) + "full title keywords order"
   The points are added to the record equivalency sum and the program exits the full title section.

Illustration 38 – `tab_cdl_mo_weights [data_tab]`

```
full title keywords                                 + 450
full title keywords order                           + 050
```

7. If the titles do not match or the word match percentage is less than 50%, then the points for "full title mismatch" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the full title section.

Illustration 39 – `tab_cdl_mo_weights [data_tab]`

```
full title mismatch                                 - 600
```

2.4.3.8 Place of publication, production, or execution Section: Field 008 position 15-17

Positions 15-17 of field 008 contain a place code.
The data is taken as is without any normalization.

1. If the place code, positions 008/15-17, contain "^^^" or "|||" (i.e. there is no place code), then no points are given and the program exits the place section.

2. If both place codes are an exact match, then the points for "country of publication exact match" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the place section.

Illustration 40 – `tab_cdl_mo_weights [data_tab]`

```
country of publication exact match                  + 040
```

3. If the place codes do not match, then the points for "country of publication mismatch" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the place section.

Illustration 41 – `tab_cdl_mo_weights [data_tab]`

```
country of publication mismatch                     - 205
```

2.4.3.9 Pagination Section: Field 300 subfield a

Field 300 subfield a contains the monograph's number of pages, volumes, etc.
The value is the highest number found in subfield a.

1. If both values are an exact match and higher than ‘10’, then the points for "300a exact match gt 10" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the pagination section.

Illustration 42 – `tab_cdl_mo_weights [data_tab]`

```
300a exact match gt 10                                  + 100
```

2. If both values are an exact match and lower than ‘10’, then the points for "300a exact match lt 10" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the pagination section.

Illustration 43 – `tab_cdl_mo_weights [data_tab]`

```
300a exact match lt 10                                  + 050
```

3. If the values do not match and the difference between them is less than ‘10’ and both are higher than ‘10’, then the points for "300a match within 10 both gt 10" in the tab_cdl_mo_weights table is added to the record equivalency sum and the program exits the pagination section.

Illustration 44 – `tab_cdl_mo_weights [data_tab]`

```
300a match within 10 both gt 10                         + 050
```

4. If the values do not match and the difference between them is less than ‘10’ and any of them is less than ‘10’, then the points for "300a match within 10 either lt 10" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the pagination section.

Illustration 45 – `tab_cdl_mo_weights [data_tab]`

```
300a match within 10 either lt 10                       + 020
```

5. In all other cases the points for "300a mismatch" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the pagination section.

Illustration 46 – `tab_cdl_mo_weights [data_tab]`

```
300a mismatch                                           - 225
```

2.4.3.10 Publisher Section: Field 260 subfield b

Field 260 subfield b contains the monograph’s publisher information.
The data undergoes normalization according to filing code 75.

Illustration 47 – `tab_filing [data_tab]`

```
75   del_subfield
75   to_upper
75   suppress
75   numbers
75   compress            '
75   to_blank            !@#$%^&*()_+-={}[]:";<>?,./~`
75   expand_num
75   non_filing
75   pack_spaces
75   char_conv           FILING-KEY-01
```

1. If field 260 subfield b is missing from one or from both records then no points are given and the program exits the publisher section.

2. If both publishers are an exact match then the points for "260b exact match" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the publisher section.

Illustration 48 – `tab_cdl_mo_weights [data_tab]`

```
260b exact match                                    + 100
```

3. If any of the publishers are contained in the other publisher, the points for "260b occur within" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the publisher section.

Illustration 49 – `tab_cdl_mo_weights [data_tab]`

```
260b occur within                                   + 100
```

4. If the publishers do not match, then the points for "260b mismatch" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the publisher section.

Illustration 50 – `tab_cdl_mo_weights [data_tab]`

```
260b mismatch                                     - 025
```

2.4.3.11 Main Entry Section: Fields 100, 110 and 111

There are three main entry fields:

- Personal name (100)
- Corporate Name (110)
- Meeting Name (111)

The fields on which a matching is performed in this section are:

- 100 a,b,c,d,k,q
- 110 a,b,c,d,k,n
- 111 a,b,c,d,e,g,k,n,q

The content of these fields is concatenated to one string on which the comparison is made.

The data undergoes normalization according to filing code 75.

Illustration 51 – `tab_filing [data_tab]`

```
75   del_subfield
75   to_upper
75   suppress
75   numbers
75   compress            '
75   to_blank            !@#$%^&*()_+-={}[]:";<>?,./~`
75   expand_num
75   non_filing
75   pack_spaces
75   char_conv           FILING-KEY-01
```

1. If both main entry fields are an exact match then the points for "main entry match" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the main entry section.

Illustration 52 – `tab_cdl_mo_weights [data_tab]`

```
main entry match                                    + 125
```

2. If all three main entry fields are missing from both records then the points for "main entry both missing" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the main entry section.

Illustration 53 – `tab_cdl_mo_weights [data_tab]`

```
main entry both missing                             + 075
```

3. If all three main entry fields are missing from one of the records, but at least one of them exists in the other, then the points for "main entry one missing" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the main entry section.

Illustration 54 – `tab_cdl_mo_weights [data_tab]`

```
main entry one missing                              + 025
```

4. If there are identical words in both main entry lines and the percentage of these words is 50% or more of the longest main entry's words and their order is not the same, then the following point calculation is made:
"main entry keywords"* percentage of matching words
The points are added to the record equivalency sum and the program exits the main entry section.

Illustration 55 – `tab_cdl_mo_weights [data_tab]`

```
main entry keywords                                    + 080
```

5. If there are identical words in both main entry lines and the percentage of these words is 50% or more of the longest main entry's words and their order is the same, then the following point calculation is made:
("main entry keywords" * percentage of matching words) + "main entry keywords order"
The points are added to the record equivalency sum and the program exits the main entry section.

Illustration 56 – `tab_cdl_mo_weights [data_tab]`

```
main entry keywords                                    + 080
main entry keywords order                              + 010
```

6. If the main entries do not match or the word match percentage is less than 50%, then the points for "main entry mismatch" in the tab_cdl_mo_weights table are added to the record equivalency sum and the program exits the main entry section.

Illustration 57 – `tab_cdl_mo_weights [data_tab]`

```
main entry mismatch                                    - 200
```

2.4.3.12 Final Monograph Evaluation Section

The points up until now are calculated into a value. This value is compared to the "treshold" line in the tab_cdl_mo_weights table.

Illustration 58 – `tab_cdl_mo_weights [data_tab]`

```
treshold                                               + 875
```

1. If the value is higher than the threshold, the records are considered a match.
2. If the value is lower than the threshold, the records are considered a mismatch.

2.4.4 Assigning weights for record matching of serials and list of common serial titles

Each step in the record matching phase of the equivalency building algorithm compares data in the base record with data in every potential candidate.

The equivalency building algorithm of monographs is based on the following MARC fields: 010, 022, 008, 245, 260, 110, 111 and 130.

2.4.4.1 LCCN Section: Field 010 subfields a,z

Field 010 holds the LC control Number.
The data undergoes normalization according to filing code 70.

Illustration 59 – `tab_filing [data_tab]`

```
70   del_subfield
70   compress              -
70   compress_blank
70   non numeric
```

Subfields ‘a’ and ‘z’ are taken into account in this match. Only the first 5 instances of subfield ‘a’ or subfield ‘z’ are compared until they fit one of the steps listed below.

1. If there is no 010 field in one or both records then no points are given and the program exits the LCCN section.

2. If there is a match between subfield a and subfield a then the points for "010aa match" in the tab_cdl_se_weights table are taken and the program exits the LCCN section.

Illustration 60 – `tab_cdl_se_weights [data_tab]`

```
010aa match                                          + 200
```

3. If there is a match between subfield a and subfield z then the points for "010az match" in the tab_cdl_se_weights table are taken and the program continues to the next match in this section.

Illustration 61 – `tab_cdl_se_weights [data_tab]`

```
010az match                                          + 100
```

4. If there is a match between subfield z and subfield z then the points for "010zz match" in the tab_cdl_se_weights table is taken.

Illustration 62 – `tab_cdl_se_weights [data_tab]`

```
010zz match                                          + 100
```

The highest number between the points given in steps 3 or in step 4 are taken and the program exits the LCCN section.
If no point were given in these steps, continue to the next step in this section.

5. If subfield a and subfield a do not match then the points for "010aa mismatch" in the tab_cdl_se_weights table are taken and the program continues to the next match in this section.

Illustration 63 – `tab_cdl_se_weights [data_tab]`

```
010aa mismatch                                       - 470
```

6. If subfield a and subfield z do not match then the points for "010az mismatch" in the tab_cdl_se_weights table are taken.

Illustration 64 – `tab_cdl_se_weights [data_tab]`

```
010az mismatch                                          - 050
```

The lowest number between the points given in steps 5 and in step 6 are taken and the program exits the LCCN section.

2.4.4.2 ISSN Section: Field 022 subfields a,z,y

Field 022 holds the Serial's ISSN.
The data undergoes normalization according to filing code 72.

Illustration 65 – `tab_filing [data_tab]`

```
72   issn
```

Subfields 'a', 'z' and 'y' are taken into account in this match. Only the first 5 instances of subfield 'a' 'z' or 'y' from all 022 fields (not for every 022 field) are compared until they fit to one of the steps listed below.

1. If there is no 022 field in one or both records then no points are given and the program exits the ISSN section.

2. If there is a match between subfield a and subfield a then the points for "022aa match" in the tab_cdl_se_weights table are taken and the program exits the ISSN section.

Illustration 66 – `tab_cdl_se_weights [data_tab]`

```
022aa match                                             + 200
```

3. If there is a match between subfield a and subfield y then the points for "022ay match" in the tab_cdl_se_weights table are taken and the program continues to the next match in this section.

Illustration 67 – `tab_cdl_se_weights [data_tab]`

```
022ay match                                             + 100
```

4. If there is a match between subfield a and subfield z then the points for "022az match" in the tab_cdl_se_weights table are taken and the program continues to the next match in this section.

Illustration 68 – `tab_cdl_se_weights [data_tab]`

```
022az match                                             + 050
```

5. If there is a match between subfield y and subfield y then the points for "022yy match" in the tab_cdl_se_weights table are taken and the program continues to the next match in this section.

Illustration 69 – `tab_cdl_se_weights [data_tab]`

```
022yy match                                          + 050
```

6. If there is a match between subfield y and subfield z then the points for "022yz match" in the tab_cdl_se_weights table are taken and the program continues to the next match in this section.

Illustration 70 – `tab_cdl_se_weights [data_tab]`

```
022yz match                                          + 030
```

7. If there is a match between subfield z and subfield z then the points for "022zz match" in the tab_cdl_se_weights table are taken.

Illustration 71 – `tab_cdl_se_weights [data_tab]`

```
022zz match                                          + 010
```

The highest number between the points given in steps 3 up to 7 are taken and the program exits the ISSN section. If no points were given, the program continues to the next match in this section.

8. If subfield a does not match subfield a, then the points for "022aa mismatch" in the tab_cdl_se_weights table are taken and the program exits the ISSN section.

Illustration 72 – `tab_cdl_se_weights [data_tab]`

```
022aa mismatch                                       - 250
```

2.4.4.3 LCCN and ISSN Weights Comparison

The points from both LCCN and ISSN sections were not added to the record equivalency sum.

The points of one of them are added to the record equivalency sum according to the following algorithm:

1. If either one of the weights is zero, the non zero value is added.
2. When one is positive and the other negative, the sum of both values is added.
3. When both are positive, the highest value is added.
4. When both are negative, the lower value is added.

2.4.4.4 Full Title Section: Field 245 subfields a,b,n,p

Field 245 contains the Serial's title.
The data undergoes normalization according to filing code 75.

Illustration 73 – `tab_filing [data_tab]`

```
75   del_subfield
75   to_upper
75   suppress
75   numbers
75   compress            '
75   to_blank            !@#$%^&*()_+-={}[]:";<>?,./~`
75   expand_num
75   non_filing
75   pack_spaces
75   char_conv           FILING-KEY-01
```

1. If the titles are an exact match and the title is not in the ‘list of common serial titles’ (see section 2.4.4.11), then the points for "full title match" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the full title section.

Illustration 74 – `tab_cdl_se_weights [data_tab]`

```
full title match                                    + 600
```

2. If the titles are an exact match and the title is in the ‘list of common serial titles’ (see section 2.4.4.11), then the points for "full common title match" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the full title section.

Illustration 75 – `tab_cdl_se_weights [data_tab]`

```
full common title match                             + 135
```

In the following steps, the mach is made only on subfields a and p:

3. If the titles (subfields a and p only) are an exact match and the title is not in the ‘list of common serial titles’ (see section 2.4.4.11), then the points for "full truncated title match" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the full title section.

Illustration 76 – `tab_cdl_se_weights [data_tab]`

```
full truncated title match                          + 175
```

4. If the titles (subfields a and p only) are an exact match and the title is in the ‘list of common serial titles’ (see section 2.4.4.11), then the points for "full truncated common title match" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the full title section.

Illustration 77 – `tab_cdl_se_weights [data_tab]`

```
full truncated common title match                   + 135
```

5. In all other cases the points for "full title mismatch" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the full title section.

Illustration 78 – `tab_cdl_se_weights [data_tab]`

```
full title mismatch                                    - 600
```

2.4.4.5 Evaluation Section

The points up until now are calculated into a value. This value is compared to the "treshold" line in the tab_cdl_se_weights table.

Illustration 79 – `tab_cdl_se_weights [data_tab]`

```
treshold                                               + 800
```

1. If the value is higher than the threshold, the records are considered a match and the match process is stopped.

2. If the value is below the threshold, the process continues to the next step in the algorithm.

2.4.4.6 Date Section: Field 008 position 07-10

Positions 07-10 of field 008 contain a date.
The data is taken as is without any normalization.

1. If the last two positions of the date, positions 008/09-10, contain one of the values: "uu", "^^" or spaces (i.e. there is no exact date) in either record, then no points are given and the program exits the date section.

2. If either date is lower than 1800 or higher than 2500 then no points are given and the program exits the date section.

3. If the dates are an exact match, then the points for "date exact match" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the date section.

Illustration 80 – `tab_cdl_se_weights [data_tab]`

```
date exact match                                       + 225
```

4. If there is a one year difference between the dates (upwards or downwards) then the points for "date within 1" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the date section.

Illustration 81 – `tab_cdl_se_weights [data_tab]`

```
date within 1                                          + 050
```

5. If there is a two year difference between the dates (upwards or downwards) then the points for "date within 2" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the date section.

Illustration 82 – `tab_cdl_se_weights [data_tab]`

```
date within 2                                    + 025
```

6. If the date's fourth digit in either record is zero, then the points for "date last digit zero" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the date section.

Illustration 83 – `tab_cdl_se_weights [data_tab]`

```
date last digit zero                             + 020
```

7. If either date is zero then no points are given and the program exits the date section.
8. In all other cases the points for "date mismatch" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the date section.

Illustration 84 – `tab_cdl_se_weights [data_tab]`

```
date mismatch                                    - 150
```

2.4.4.7 Country Code of publication, production, or execution Section: Field 008 position 15-17

Positions 15-17 of field 008 contain a country code.
The data is taken as is without any normalization.

1. If the place code, positions 008/15-17, contain "^^^" or "|||" (i.e. there is no place code), then no points are given and the program exits the place section.

2. If both country codes are an exact match then the points for "country of publication exact match" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the country section.

Illustration 85 – `tab_cdl_se_weights [data_tab]`

```
country of publication exact match               + 040
```

3. If the country codes do not match, then the points for "country of publication mismatch" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the country section.

Illustration 86 – `tab_cdl_se_weights [data_tab]`

```
country of publication mismatch                  - 020
```

2.4.4.8 Place of Publication Section: Field 260 subfield a

Field 260 subfield a contains the serial's place of publication.
The data undergoes normalization according to filing code 75.

Illustration 87 – `tab_filing [data_tab]`

```
75   del_subfield
75   to_upper
75   suppress
75   numbers
75   compress            '
75   to_blank            !@#$%^&*()_+-={}[]:";<>?,./~`
75   expand_num
75   non_filing
75   pack_spaces
75   char_conv           FILING-KEY-01
```

1. If field 260 subfield a is missing from one or both records then no points are given and the program exits the place section.

2. If both places are an exact match then the points for "260a exact match" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the place section.

Illustration 88 – `tab_cdl_se_weights [data_tab]`

```
260a exact match                                    + 200
```

3. In all other cases the points for " 260a mismatch" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the place section.

Illustration 89 – `tab_cdl_se_weights [data_tab]`

```
260a mismatch                                       - 100
```

2.4.4.9 Main Entry Section: Fields 110, 111 and 130

There are three main entry fields:

- Corporate Name (110)
- Meeting Name (111)
- Uniform Title (130)

The fields on which a matching is performed in this section are:

- 110 a,b,c,d,k,n
- 111 a,b,c,d,e,g,k,n,q
- 130 a,d,g,k,l,m,n,o,p,r,s,t

The content of these fields is concatenated to one string on which the comparison is made.

The data undergoes normalization according to filing code 75.

Illustration 90 – `tab_filing [data_tab]`

```
75   del_subfield
75   to_upper
75   suppress
75   numbers
75   compress             '
75   to_blank             !@#$%^&*()_+-={}[]:";<>?,./~`
75   expand_num
75   non_filing
75   pack_spaces
75   char_conv            FILING-KEY-01
```

1. If all three main entry fields are missing from one or both records then no points are given and the program exits the main entry section.

2. If both main entry fields are an exact match then the points for "main entry match" in the tab_cdl_se_weights table are added to the record equivalency sum and the program exits the main entry section.

Illustration 91 – `tab_cdl_se_weights [data_tab]`

```
main entry match                                    + 200
```

3. If there are identical words in both main entry lines and the percentage of these words is 60% or more of the longest main entry's words and their order is not the same, then the following point calculation is made:
   main entry keywords"* percentage of matching words
   The points are added to the record equivalency sum and the program exits the main entry section.

Illustration 92 – `tab_cdl_se_weights [data_tab]`

```
main entry keywords                                 + 075
```

4. If there are identical words in both main entry lines and the percentage of these words is 60% or more of the longest main entry's words and their order is the same, then the following point calculation is made:
   ("main entry keywords" * percentage of matching words) + "main entry keywords order"
   The points are added to the record equivalency sum and the program exits the main entry section.

Illustration 93 – `tab_cdl_se_weights [data_tab]`

```
main entry keywords                                 + 075
main entry keywords order                           + 025
```

5. If the main entries do not match or the word match percentage is less than 60%, then the points for "main entry mismatch" in the tab_cdl_se_weights

table are added to the record equivalency sum and the program exits the main entry section.

Illustration 94 – `tab_cdl_se_weights [data_tab]`

```
main entry mismatch                                    - 250
```

2.4.3.10 Final Serial Evaluation Section

The points up until now are calculated into a value. This value is compared to the "treshold" line in the tab_cdl_se_weights table.

Illustration 95 – `tab_cdl_se_weights [data_tab]`

```
treshold                                               + 800
```

3. If the value is higher than the threshold, the records are considered a match.
4. If the value is lower than the threshold, the records are considered a mismatch.

2.4.4.11 List of common serial titles

The list of common serial titles is consulted during the matching stage of the CDL equivalency building process. Title matches in serial records that have titles that are considered "common" receive significantly fewer points than title matches for other serial records (135 instead of 600). The table lists "common" titles in a given database or Union Catalog. If the contents of the table change, `p-union-02` should be rerun.

Illustration 96 - `tab_com_tit_cdl [data_tab]`

```
ANNUAL REPORT
ANNUAL REPORT FOR
ANNUAL REPORT FOR THE FISCAL YEAR ENDED MARCH
BIENNIAL REPORT
BULLETIN
CALENDAR
CATALOGUE
CIRCULAR
```

2.4.5 Additional table settings

There are two additional settings that must be in place for the equivalency building processes to work. The first setting is in `tab_expand`. `Tab_expand` has a section that lists expand programs that are to be called by the two equivalency building programs, `p_union_02` and `p_union_04`. Note that both `p_union_02` and `p_union_04` use the UNION-02 section. These settings should not be changed.

Illustration 97 - `tab_expand [data_tab]`

```
UNION-02   expand_doc_extract
UNION-02   expand_doc_ntl
```

The second table that needs to be set up is `tab_filing`. Five filing routines are used by the equivalency building processes, they are: 70-75. Routine 74 is used to normalize the short title (first 25 characters) for the candidate phase of the routine, routine 75 is used to normalize all text fields during the match phase of the routine, routine 70-73 are used to normalize numeric fields (010,020 and 022) for indexing and during the match phase of the routine.

Illustration 98 - `tab_filing [data_tab]`

```
!* The following six routines (70, 71, 72, 73, 74, 75) are needed
for !* Union Catalog and Union Catalog
!* 70 - LCCN normalization for LCCN index to be used by Equivalency
!* building program and union_match_cdl_mo
70   del_subfield
70   compress             -
70   compress_blank
70   non_numeric
!* 71 - ISBN normalization for ISBN index to be used by Equivalency
!* building program and union_match_cdl_mo
71   isbn
!* 72 - ISSN normalization for ISNN index to be used by Equivalency
!* building program and union_match_cdl_se
72   issn
!* SID index
73   del_subfield
73   compress             -
73   compress_blank
73   cut_prefix
!* Reserved for equivalency building program - used to create the
!* normalized title (NTL)
74   del_subfield
74   non_filing
74   to_blank             !@#$%^&*()_+-={}[]:";<>?,./~`
74   char_conv            FILING-KEY-01
74   compress_blank
74   first_25
74   to_lower
!* Reserved for equivalency building program - used to normalize
!* all text fields in the record during the match phase of
!* equivalency building
75   del_subfield
75   to_upper
75   suppress
75   numbers
75   compress             '
75   to_blank             !@#$%^&*()_+-={}[]:";<>?,./~`
75   expand_num
75   non_filing
75   pack_spaces
75   char_conv            FILING-KEY-01
```

2.5 Phase Three - Selecting the preferred record

The third phase of the equivalency building process is selecting the preferred record. The program for selecting preferred records is identified in the union_global_param table in alephe_tab and the routine for selecting preferred records is configured in a

single configuration table, union_preferred. Currently there is only one program available for preferred record selection, it is union_preferred_cdl.

2.5.1 Server table

The following line should be entered in the server table `union_global_param`. This table controls the individual components of the equivalency algorithm.

Illustration 99 – `union_global_param [alephe_tab]`

```
CUN01 B preferred_prog       union_preferred_cdl
```

2.5.2 Library tables

The table `union_preferred`, located in the data_tab directory of your library, configures the basis for selecting a preferred record from a set of equivalent records. It does so by assigning points for field presence, and/or subfield or fixed field values. After each record in a set of equivalent records gets a weight, the record with the greatest weight becomes the preferred record.

Illustration 100 – `union_preferred [alephe_tab]`

```
!!!!!-!!!!!!-!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!-!!!
LDR   F05-01 EQUAL      d                            -10
LDR   F17-01 NOT-EQUAL  1,2,3,4,5,7,8,u,z            001
040   a      EQUAL      DLC                          003
100##        PRESENT                                 001
110##        PRESENT                                 001
111##        PRESENT                                 001
130##        PRESENT                                 001
24###        PRESENT                                 001
6####        PRESENT                                 001
700##        PRESENT                                 001
710##        PRESENT                                 001
711##        PRESENT                                 001
730##        PRESENT                                 001
800##        PRESENT                                 001
810##        PRESENT                                 001
830##        PRESENT                                 001
880##        PRESENT                                 001
```

2.6 Building equivalent record tables

The three steps described above, candidate record selection, record matching and preferred record selection, are performed during equivalency building by the batch processes p_union_02 and p_union_04. In addition to those two basic processes, there are a number of other processes that must be run.

2.6.1 Batch Processes

There are a total of five processes for building and maintaining record equivalencies.

1. **`p_union_01`** – Run after initial conversion and data load. Builds Z120 record for each bibliographic record. The parameter is: database name. After p_union_01 each record is equivalent to itself but not to any other record.

   $aleph_proc/csh –f p_union_01 USM90

2. **`p_union_02`** – Run after database has been indexed. Can be run with multiple processes. Parameters are database name, start number, end number, number of processes, and mode - batch/test. This process locks the library. P_union_02 and should always be run on the entire database; it should not be run on parts of a database. Doing so results in inconsistencies in the record equivalency tables.

   Prior to version 16.02 there was a parameter - rebuild links.

   $aleph_proc/csh –f p_union_02 USM90,000000000,999999999,B

3. **`p_union_04` –** Ongoing process, should be run in daily mode or each time records have been loaded using p_union_13. This process rebuilds record equivalencies for records that have been updated [identified by a Z120 record flagged N for update.] From version 16.02 and on p_union_04 is started by p_union_13 unless it is already running; it no longer has to be restarted each night.

4. **create_z127 –** From version 15.2 and later, it is necessary to run this process to create Z127 records after running p_union_02 to create Z120 records. The process does not have any parameters – the syntax for running it is: csh –f $aleph_proc/create_Z127

5. **load_z127_to_mem –** After creating Z127 records they need to be loaded into the memory. The syntax for running this job is: csh –f $aleph_proc/ load_z127_to_mem

2.6.2 Workflow

After bibliographic records have been loaded p_union_01 should be run to create empty equivalency tables. Then, build indexes for the database (two processes are required - p_manage_01, p_manage_05), and after indexes have been built, run p_union_02 and p_union_04.

2.6.3 Differences between p_union_02 and p_union_04

P_union_04 runs all the time and is responsible for maintaining equivalencies for new and updated records. There are a number of important differences between p_union_02 and p_union_04. They are:

- P_union_02 rebuilds all equivalences for the range of records entered
- P_union_04 only rebuilds equivalency records that are flagged N
- P_union_02 builds equivalency records one by one and thus does not maintain symmetry between equivalency tables

- P_union_04 builds an equivalency record and updates all related records thus maintaining symmetry between records
- P_union_02 can run with multiple processes
- P_union_04 has a single process
- P_union_02 only attempts to build equivalencies for records with fewer than 100 candidates. If a record has too many candidates, the Z120 is flagged N and p_union_04 attempts to build it
- P_union_04 attempts to build equivalencies for all records – even those with many candidates

Note that libraries should never run p_union_02 on part of a database specifically because it does not maintain symmetry. Only the p_union_04 process should be used to update a portion of the database.

2.6.4 Miscellaneous tables

The variable in `prof_library` that controls the `p_union_02` loop length is:

Illustration 101 – `prof_library [data_root]`

```
15.2      setenv union_loop_length  5000

 16       setenv p_union_02_loop_length    5000
```

setenv union_loop_length 10000

For more information on setting loop lengths, see the document entitled, “How to Run Index Jobs”.

2.6.5 Triggers

Union Catalog libraries need to have a trigger for the Z127 table set up and active. The trigger sees to it that the contents of the Z120 table and the Z127 table stay in synch. Note that the contents of the Z127 table are derived from the Z120 table. There are two processes that create a load the Z127 table after a library has been loaded and the Z120 table built. The Oracle trigger takes over from there and continues to update existing Z127 records and add new Z127 records as required. Use the UTIL/A/17 menu to check the status of your Z127 trigger.

3 OPAC Functionality

ALEPH libraries that are defined as Union Catalog libraries can offer bases that are “union catalog enabled” and bases that are not. Union Catalog enabled bases are bases that use the record equivalency tables to de-duplicate result sets and merge records. Non-union catalog enabled bases do not provide de-duplication or record merging. Union Catalogs wishing to offer individual campus views can set up logical bases are comprised of records from a single campus. If the base is not ‘union catalog enables’, only that campus’s records display. For more information on logical bases, see the document “How to set up bases and filters in ALEPH”.

3.1 Enabling Union Catalog bases in your OPAC

Logical bases are set in the table tab_base.lng which is located in alephe_tab. In libraries that are set up as Union Catalog or Union View libraries in tab100, bases that begin with U- are Union Catalog bases, and bases that do not begin with a U- are standard bases. Note that the only difference between Union Catalog bases and all other bases is that the result set in Union Catalog bases uses record equivalency tables. They are identical in all other ways. Thus, be sure to set all columns in tab_base.lng and be sure to run the same processes as are run for any logical base (p_manage_32). See ALEPH documentation for more information on Logical bases.

3.2 Configuring Your OPAC Interface for Union Catalog

3.2.1 Union Catalog settings in www_server.conf

The following setting in www_server.conf applies to Union Catalog bases in version 14.2 and 15.5. This setting was removed from version 15.2, 16.02 and all subsequent versions:

Illustration 102 - `www_server.conf [alephe_root]`

```
setenv www_sort_limit                800
```

The sort limit is also used as the de-duplication limit. Only this number of records are de-duplicated. This means that if a retrieval set is larger than the limit, only part of the retrieval set is de-duplicated. The de-duplicated records are the first to be displayed.

In versions without this setting [version 15.3 and higher], the system has no limit on the retrieval set size, but there is a system wide limit of 1,000 records for display and sorting. In a Union Catalog, up to 1,000 de-duplicated and merged records display. However, because the entire set is not de-duplicated, but rather only the number needed to reach a display of 1,000 records, the set size does not accurately reflect the number of de-duplicated records.

3.2.2 Merged display

The system uses the configuration table `tab_merge_union` located in the `data_tab` directory of the bibliographic library to configure the merged display. This table controls record merging for display and allows you to set, on a tag-by-tag basis, which fields are displayed from the preferred and non-preferred records.

Illustration 103 - `tab_merge union [data_tab]`

```
!1 2 3          4
!!-!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!
90 1 Y #####
90 1 N SID##
90 1 N 852##
90 1 N 856##

90 2 Y 5050#,u,*
90 2 Y 852##
```

```
90 2 Y 856##
```

Key to the table:

Column 1 – Merge set –always is 90 for Union Catalog merge.

Column 2 – Merging direction (1 refers to preferred record, 2 to each equivalent record).

Column 3 – Action (Y, N, C) Y – retain field, N – delete field, C – retain field if it is original).

Column 4 – Field tag.

Column 4 (continuation after ,) –subfield and contents to match on. Example: 01 2 Y 590##,5,*abc* (meaning "if there is a subfield 5 which contains 'abc').

Note: The program `expand_doc_merge_union` must be called in `tab_expand` for each instance where a merged display is required.

For example, a Union Catalog site might want to have a merged display in the Web OPAC but not the Z39.50. In this case, the WEB-FULL section of `tab_expand` would call expand_doc_merge_union but the Z39_SERVER would not.

Illustration 104 – `tab_expand [data_tab]`

```
WEB-FULL    expand_doc_merge_union
```

3.2.3 Displaying individual records

In addition to displaying merged records that contain fields from the preferred record and from all equivalent records, it is also possible to display individual, unmerged records from each contributing library. Display of the individual unmerged record is activated by the SID2 link and the display itself is configured in the DIRECT-DOC section of tab_expand.

Illustration 105 - edit_doc_999.eng [data_tab]

```
## SID2                  D LSource Record (SID2)Y       E
```

Illustration 106 – tab_expand [data_tab]

```
WEB-DIRECT expand_doc_direct
WEB-DIRECT expand_doc_union_holding       DIRECT=Y,SORT=1,JUMP-TO=Y
```

The link is hard-coded to display the Library name from the z124_campus_name field. For more information about the Z124 table, see section 4 Embedded Holdings. Selecting the hyperlinked library name results in the display of that library's record and holdings, while selecting the hyperlinked "global view" returns you to the merged display of all records and all holdings.

Illustration 107 – OPAC full view of a merged record

3.2.4 Displaying Libraries Which Own the Record

To display the list of libraries in the brief list which own the record, add the following line to ./[Union Catalog lib]/tab/www_tab_short.{lng}.

4 L Source	S	## SID

4 Embedded Holdings

In a Union Catalog, holdings statements are stored in 8XX fields embedded in the bibliographic record; this is different from a non-Union Catalog library where location information is stored in MARC 21 holdings records that are stored in a separate but linked library. A bibliographic record can have multiple holdings fields embedded in a single bibliographic record, and all of the fields display. In addition to displaying the holdings location, the Union Catalog also displays electronic links and links to real-time circulation status.

Real time circulation status is obtained from ALEPH libraries using the Ex Libris X-Server or the Z39.50 server. The X-Server needs to be licensed on both the Union Catalog server and on the local library's server. X-server functionality is available from version 14.2 and on.

4.1 Location table

Typically, the embedded 852 field contains a campus or institution code as well as a library code and call number. The mechanism for translating location codes into location statements is as follows. A configuration table, tab_locations, stores a list of campus and library codes and their corresponding names. This table is loaded into Oracle as the Z124 table. The system matches the contents of the SID $$b and the 852 $$a and $$b against the key of the Oracle table. If a match is found, the location names display instead of the codes. If no match is found, the location does not display

in the OPAC. Note that more information about the SID field is available in section 5.2.3 Mandatory Fields.

4.1.1 Tab_locations [data_tab]

Enter location information in tab_locations table in the data_tab directory of the Union Catalog library (xxx90).

The table has ten columns. The first, the location key, is a 24 character field that contains a location code plus a sublocation code and matches on the combined contents of subfields a and b of embedded 852 fields. The remaining nine columns contain data useful for display and/or indexing. The remaining fields are:

- campus type
- campus region
- suppress display
- campus code
- campus base
- primary location
- full campus name
- call number prefix
- notes

All of the data elements and their lengths are listed in the table below. Data elements marked (I) are expanded into the bibliographic record for indexing. Data elements marked (D) are included in the location display on the full record display in the Web OPAC, and data elements marked (R) are included in the results list in the Web OPAC .

Length	Data Element
24	Location key
1	Type (I)
1	Region (I)
1	Suppress display (values are Y or N or blank – blank is No)
5	Campus code (I)(R)
15	Campus base (I)
9	Primary location (D)
50	Campus name (D)
15	Call number prefix (D)
70	Notes (D)

Once the table tab_locations has been filled out, it needs to be loaded into Oracle. This is done by running the process p_load_z124. This process can only be run from the command line.

4.1.2 Oracle tables

The Z124 table stores data from the location table. The following illustration shows the structure of the Z124 table:

Illustration 108 – Z124 table

```
          02 Z124-REC-KEY              PICTURE X(24).                        K
          02 Z124-TYPE                 PICTURE X(1).
          02 Z124-REGION               PICTURE X(1).
          02 Z124-SUPPRESS             PICTURE X(1).
          02 Z124-CAMPUS-CODE          PICTURE X(5).
          02 Z124-AT                   PICTURE X(15).
          02 Z124-PRIMARY-LOCATION     PICTURE X(9).
          02 Z124-CAMPUS-NAME          PICTURE X(50).
          02 Z124-CALL-NUMBER          PICTURE X(15).
          02 Z124-NOTES                PICTURE X(70).
          02 Z124-SORT-KEY             PICTURE X(50).
```

There is a single process, p_load_z124, which is used to load the Z124 table. It takes the table tab_locations and loads it into Oracle. Note that this must be done each time tab_location is changed. The table can be located in data_tab, data_files or data_scratch. The second parameter of the process is table name If the table is in data_files, run it with name only. If the table is located in data_scratch or data_tab the process must include a directory relative to data_files as shown below:

Table in data_files: csh -f $aleph_proc/p_load_z124 usm90,z124,n,n,n,n
Table in data_tab: csh -f $aleph_proc/p_load_z124 usm90,../tab/tab_locations,z124,n,n,n,n
Table in data_scratch: csh -f $aleph_proc/p_load_z124 usm90,../scratch/tab_locations,z124,n,n,n,n

4.1.3 Indexing and Displaying Locations

Four of the ten location fields can be expanded into bibliographic records either for indexing or for display. They are: type, region, campus code, and campus base. These fields are particularly useful for creating logical bases. A single expand program, expand_doc_cdl_location, expands the indexable fields into the bibliographic record as four separate fields, CMP1-4. Mapping is as follows:

Type	CMP1
Region	CMP2
Code	CMP3
Campus Base (z124_AT)	CMP4

Add expand_doc_cdl_location to the WORD section of tab_expand and the following lines to tab11_word to create keyword indexes on locations.

Illustration 109 - tab_expand [data_tab]

```
WORD       expand doc cdl location
```

Illustration 110 - tab11_word [data_tab]

```
CMP1#                                          03 B   WRD  WC1  WID
CMP2#                                          03 B   WRD  WC2  WID
CMP3#                                          03 B   WRD  WC3  WID
CMP4#                                          03 B   WRD  WC4  WID
```

4.2 Displaying Embedded Holdings

Display of embedded holdings in the Web OPAC is controlled by one of two programs, expand_doc_union_holding and expand_doc_union_holding_cdl. The two programs are very similar; however, expand_doc_union_holding_cdl has some institution specific behavior programmed in.

4.2.1 expand_doc_union_holding

This program is called in tab_expand and controls the holdings section of the Union Catalog full record display. The program generates a virtual field called HOL from data in the Bibliographic record, in the Oracle location table (Z124) and in database configuration tables. An HOL field is created for each 852 or 856 field in a bibliographic record. The HOL field displays in the web OPAC as well as in the terminal mode client.

The program works as follows:

1. The expand_doc_merge_union program merges equivalent records, taking holdings fields (SID, 852 and 856) from each equivalent record.
2. Each 852 $$a + $$b in a bibliographic record is matched against the Oracle location table (z124). An HOL field is created for each field that has a matching location tables. Note that if no match is found, no HOL line is generated.
3. Fields from Oracle location table are concatenated with 852 field from the bibliographic record as follows:
    - HOL $$a => Source tag – tag in the bibliographic record that data is taken from
    - HOL $$b => Location, campus name - taken from z124_campus_name
    - HOL $$c => Sub-location, library name – taken from z124-primary_location
    - HOL $$d => Call number - taken from the following places in this order
        1. 852 $$k
        2. z124_call_number
        3. 852 $$h $$i $$j
    - HOL $$e => Availability – This section is further controlled by the parameter “JUMP-TO” as explained below. If the source tag is 852 then the availability is circulation status. There is more information about configuring this section below. If the source tag is 856 then the availability is online and 856 $$u is used to generate a link.
    - HOL $$f => Note – taken from the following list of subfields in the order listed. 852 $$g $$p $$3* $$m z124_notes 852 $$z $$D**.
    - HOL $$g => Campus code – taken from z124_campus_code

4. Three parameters control behavior of the HOL display. The three parameters are:

   - DIRECT – controls if the program is called when a single un-merged record is displayed. Parameters are Yes and No.
   - SORT – controls the sort of the HOL fields. One of two sort routines can be used:

     1 – Sort using the location table sort key (z124_sort_key). The z124_sort key is filled in when the contents of tab_locations are loaded into Oracle and each line is assigned a running number based on position; the first line in the table is 1, the second is 2 etc.

     2 – Sort alphabetically by Campus Name.

     In addition, a sub-sort is available: “a”: It can be added to the existing sort types. If the sort type is 1a, holdings displayed in the full view are sorted first according to z124_sort_key, then by $$8, which causes the Electronic locations (856 fields) to appear right after their related 852 field.

     If the sort type is 2a, holdings displayed in the full view are first sorted according to alphabetical order by Campus Names and then by an inner arrangement according to $$8.

   - JUMP-TO – controls creation of a link from the Holdings display for circulation status and for electronic locations. If Jump-to is set to Yes then a link to circulation status is created for HOL fields created from 852 fields, and a link to electronic locations is created for HOL fields created from 856 fields.

     *852 $$3 – subfield 3 of the 852 field can be used to store textual holdings statements for serials. Subfield 3 can repeat.

     **852 $$D – subfield D of the 852 field can be used to store receipt status.

4.2.2 expand_doc_union_holdings_cdl

The expand_doc_union_holdings_cdl program functions very much like the expand_doc_union_holdings program with a few minor differences. The differences are hard coded and therefore are not configurable.
The differences are as follows:

- There is special treatment for holdings locations that have a campus code that contains the code RLF. These locations are used as remote storage facilities and electronic links do not display from records that do not have any code other than RLF.

- Suppressed locations – locations that are flagged for suppression but have an electronic link, display with a link to the electronic location.

- There is no HOL subfield f. Instead, data is mapped to subfields h, i, j, and k of the HOL field, each subfield displays on a separate line in the Notes column of the location display. Note that in order to display the four separate lines, the placeholder “$0600” should be used instead of

"0500" in the fourth column of the HTML file full-set-body-1. The HTML file is located in the directory www-f-eng.

- HOL $$h => z124_notes
- HOL $$i => 852 $3 and 852 $$m
- HOL $$j => 852 $g and 852 $p
- HOL $$k => 852 $z and 852 $D

4.3 Circulation Status

The Union Catalog supports the display of up-to-date circulation status information by querying the local catalog and displaying the results in the Union Catalog. The initial query is initiated by selecting the hypertext link marked "availability" from the full display. The link displays information from the full display of the merged record and the full display of the unmerged record. The link is activated by setting the JUMP-TO variable in tab_expand to Yes. The link only works if the records in the Union Catalog point to specific records in the local catalog and if the local catalog is available. The pointer from the Union Catalog to the local catalog is the SID field. More information about the SID field is available in section 5.2.3 Mandatory Fields. If the library is not available, location information from the embedded holdings displays with a message saying that the library is not available.

The link to circulation status can be configured to use the Z39.50 server or the Ex Libris X-server.

4.3.1 Using Z39.50

More information about the Z39.50 setup can be obtained directly from Ex Libris. Bear in mind that in order for the link to circulation status to work via Z39.50, both the Union Catalog and each of the local catalogs need to have a Z39.50 server and gate set up.

4.3.2 Using the X-Server

Union Catalogs that target ALEPH libraries can use the Ex Libris X-Server instead of Z39.50 to obtain circulation status information from the local libraries. In order to use the X-Server to obtain circulation status information, it must be licensed and set up on both the Union Catalog and the local library server.

4.3.3 Union Catalog setup

The method of obtaining circulation status is set in the directory alephe_gate. There, a configuration table should be created for each campus that needs to be accessed. The convention for naming the configuration table is local library code .conf where local library code is the code that appears in the 852 field subfield a. An example of a configuration table name is bin.conf

Variables that need to be set in this table are:

1. Base – name of the base that is being targeted
2. Host name – IP of the server that is being targeted

3. Circulation program – name of the program used to display circulation status. There is currently only one program – www_f_union_circ_www_x
4. Access method – method of accessing the targeted server and base, for X-Server should be set to X-SERVER.

Illustration 111 – usm.conf [alephe_gate]

```
Z58-BASE-REMOTE          USM01
Z58-HOST-NAME            10.1.235.11:8991
Z58-CIRC-PROGRAM         www_f_union_circ_www_x
Z58-ACCESS-METHOD        X-SERVER
```

4.3.4 Local library setup

The following tables need to be configured on the server of the library being targeted in order for the x-server to work:www_x_func, user_function.eng, license.www_x, tab_z30_sort, www_server.conf

- www_s_func [alephe_tab] needs to have a line added for the function CIRC-STATUS. The line appears as follows:

########## CIRC-STATUS www_x_circ_status

- user_function.eng [alephe_tab] needs to have a line added for Circulation status. The line appears as follows:

  WWW-X L X-SERVER Interface CIRC-STATUS L Circ status

- License.www_x [alephe_tab] needs to have the X-Server enabled for circulation status. The line appears as follows:

  CIRC-STATUS Y

- tab_z30_sort [data_tab in the bibliographic library] A sort routine needs to be defined for sorting items retrieved by the X-Server. The line appears as follows:

  WWW-X A 01 A 05

- www_server.conf [alephe_root] an environmental variable needs to be added for limiting the number of items that can be retrieved by the X-Server. The line appears as follows:

  www_no_items_display 1000

- In versions 16.02 and higher, a staff user needs to be set up for the X-Server. The user needs to be set up in the local library. The username and password for this user must be WWW-X-CIRC/WWW-X-CIRC and it must be assigned

permission for the x-server interface and for displaying circulation status. Lower versions do not check for user name and password.

5 Loading records into a Union Catalog

Union Catalogs typically receive regular batches of records from each of their member libraries, these records need to be loaded into the Union Catalog. The process of loading the records into the Union Catalog has two steps, the first step involves data conversion, enrichment, and validation, and the second step involves loading and indexing the records into the Union Catalog. While the second step, the process of loading records, is identical for all libraries, the first step, where records are converted from the local format into the Union Catalog format and character set, might be slightly different for each library.

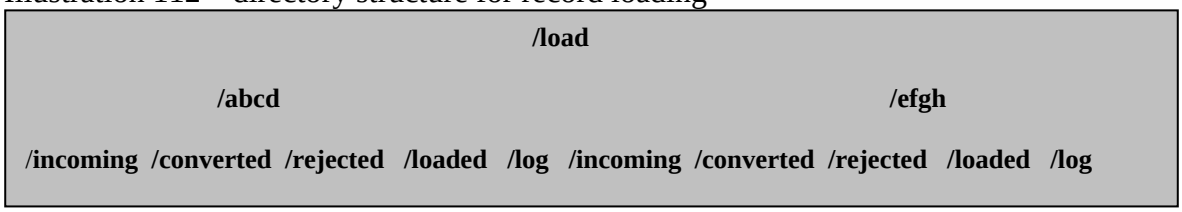
Records are placed in the incoming directory in MARC 21 format and converted from MARC 21 format into ALEPH sequential. At the same time, they are converted into UTF-8. Once in ALEPH sequential format, record conversion programs can be used to convert or enrich the data. Finally, records are validated and either moved to the converted directory for loading or rejected.

5.1 Setup and workflow:

5.1.1 Directory structure:

ALEPH Union Catalogs [XXX90] contain a directory called load and this directory contains a subdirectory for each input stream. The data_load directory is configured in prof_library. An input stream is any contributing library or organization that had distinct conversion needs. Each input stream's directory has the following five sub-directories; incoming, converted, rejected, loaded, and log. Note that an input steam name must be one to five characters.

Illustration 112 – directory structure for record loading



5.1.2 Configuration tables:

The ALEPH configuration tables that control record conversion and validation are all located in the data_tab directory of the Union Catalog [xxx90/tab]. They are tab_fix, check_doc, check_doc.eng and check_doc_mandatory. Each input stream can have a corresponding section in any one or all of these tables. The section name and the directory name are the same.

5.2 Record Conversion and pre-processing

5.2.1 Workflow

Files of incoming records in MARC 21 format are placed in the incoming directory of the appropriate input stream. The process p_union_03 picks the records up from there, convert them into ALEPH Sequential format, validates them, and moves them to one of two directories, converted or rejected. The logfile of the p_union_03 process is written to the log directory. Note that in addition to converting records from MARC format to ALEPH Sequential format, the p_union_03 process can also convert the character set of the records from MARC-8 to UTF-8. The third parameter of the p_union_03 process controls character conversion. If the records need to be converted to UTF-8, the third parameter should be set to Y. Otherwise run it with N or blank.

5.2.2 Record Conversion:

There are three stages to record conversion, they are:

- Conversion from the incoming format into ALEPH sequential
- Field-by-field data conversion or enrichment
- Record validation

During the record conversion stage a file of MARC records is converted into ALEPH sequential (p_file_01 and p_file_02). There are no configuration tables consulted at this stage. Once the records are in ALEPH sequential format, the records are “fixed”. This process is regulated by tab_fix, an ALEPH configuration table located in the data_tab directory of the Union Catalog [xxx90/tab]. Tab_fix should include a section for each input stream and can be used to call “fix” programs that are specific to the input stream as well as generic ALEPH “fix” programs. Libraries can write their own fix programs or they can use any of the pre-existing ALEPH fix programs. In addition, they can create scripts for fixing records using the ALEPH generic fix mechanism. Finally, libraries can write their own fix programs and call them in tab_fix. A list of generic fix_doc programs can be found in the database management guide. Note that fix_doc programs are entered in the table in the order in which they are to be performed.

Illustration 113 - tab_fix [data_tab]

```
UCLA    fix_doc_ucla
UCLA    fix_doc_sid_cdl
```

Key to the table:

Column 1 – Routine name/input stream
Column 2 – Program name
Column 3 – Program arguments

5.2.3 Mandatory Fields

In order for records to be considered “valid” for the Union Catalog, they must have a 005 field and an SID field. The 005 field is the MARC 21 field for storing latest transaction date, the programs that load records into the Union Catalog consults the

005 field to confirm that the record being loaded is more recent than the version of that record already in the database. The SID field is an ALEPH proprietary field. It serves as a pointer from the record in the Union Catalog to the same record in a local catalog. In addition, the SID field is used to uniquely identify records in the Union Catalog for the purpose of updating them. The structure of the SID field is $$b Local library code $$c system number in local library. Here is an example of an SID field: SID $$bUCSB$$c000034928.
If records do not already have these fields, tab_fix should be used to assign them.

5.2.4 Record Validation

A second ALEPH configuration table, check_doc, calls record validation routines for each input stream. The check_doc table works in conjunction with check_doc.eng, check_doc_doc and check_doc_mandatory. Input streams can use location specific validation programs as well as generic routines. Check_doc configures the combination of specific and generic routines that are performed on records within each input stream while check_doc_doc contains a list of fields that should be present in all incoming records. Note that each validation routine has an error code associated with it; the severity of the error and the text of the error message are set in check_doc_mandatory and check_doc.eng respectively.

Illustration 114 - check_doc [data_tab]

```
UCLA            check_doc_doc
UCLA            check_doc_ucla
```

Key to the table:

Column 1 – Routine name/input stream
Column 2 – table name/program name

Illustration 115 - check_doc_doc section 1[data_tab]

```
OC XX 5002 01 01 245##
OC XX 5008 01 99 852##
```

Key to part one of the table:

Column 1 – Section ID [OC = Occurrence]
Column 2 – Record format [BK, SE, and so on]
Column 3 – Error message code
Column 4 – Minimum number of occurrences
Column 5 – Maximum number of occurrences
Column 6-10 – Fields that must occur

Illustration 116 – check_doc_doc section 2 [data_tab]

```
D  BK 7003 2450#   Y 1####   N
D  BK 7004 2451#   Y 1####   Y
```

Key to part two of the table:

Column 1 – Section ID [D = Dependency]
Column 2 – Record format [BK, SE, and so on]
Column 3 – Error message number
Column 4 – Field code
Column 5 – Type of dependency [Y- present, N- not present]
Column 6 – Second field code
Column 7 – Type of dependency [Y- present, N- not present]

Illustration 117 - check_doc.eng [data_tab]

```
5008 L Required 852 field is missing.
5009 L 901 field subfield "c" should be "LAD".
```

Key to the table:

Column 1 – Error message number
Column 2 – Alpha (should always be L)
Column 3 – Error message text

Illustration 118 - check_doc_mandatory [data_tab]

```
UCLA        5008 M Required 852 field
UCLA        5009 M 901 $$c must be "LAD"
```

Key to the table:

Column 1 – Routine name
Column 2 – Error message number
Column 3 – Error type [M= record cannot be loaded, T=record is loaded and has ERR field]

5.2.5 Batch processes

The batch process that converts and validates records is called p_union_03. The process can be run with the following parameters: Active library, input stream, and convert to UTF-8. If an input stream is specified then the process only checks the incoming directory for that stream. If no input stream is specified, the process checks all incoming directories. Note that the list of allowed input streams vary from installation to installation.

5.3 Record Loading

Once records have been converted, enriched, and validated, they are ready to be loaded into the Union Catalog. Valid records are picked up from the subdirectory converted and loaded into the Union Catalog. The process that loads records determines if the record is new or if it already exists in the database by searching the database for the SID of the incoming record.

- If the record exists
    - The dates of the two records are compared and the most recent record is

retained. Date is taken from the 005 field.
    - If the most recent version of a record is a delete record, the record in the Union Catalog is deleted and all access points removed. Delete status is set in LDR position 6.
- If the record is new, it is loaded into the database as a new record and indexed.

Note that this entire process is hard coded. Libraries must have an SID field in their incoming and database records and an SID index set up as a direct index in tab11_ind. The index should be called SID.

5.3.1 Batch processes

The batch processes that perform the record loading and indexing are p_union_13 and p_union_14.

P_union_13 takes over from p_union_03 and loads records from the directory converted. Records are placed in /converted by p_union_03. After loading the records, the input file is moved to the directory loaded and a suffix loadout is added to the file name. A log file is created in the log directory. It logs the number of records that were added as new, the number of records that replaced existing records in the database, and the number of records that were rejected. Records are rejected if their date is earlier then the date of the same record in the database.

The p_union_13 has three parameters. They are: library name, input stream, and load equal. If no input stream is specified, the process checks all input streams. If load equal is set to Yes, incoming records that have a date equal to the date in the database record are loaded. If it is set to No, incoming records with a date equal to the database record date are rejected.

In addition to loading incoming records, p_union_13 partially indexes them and updates the record equivalency tables (Z120). The update to the equivalency tables works as follows:

- For new records, a new table is created, and the table is flagged Z.
- For updated records, the Z120 is emptied out so that the record itself is no longer equivalent to any other records in the database. In addition, the record is taken out of any record equivalencies that it occurs in, either as an equivalent or as a preferred record. Finally, the record's record equivalency table is flagged Z.

The p_union_13 process starts two related processes p_union_14 and p_union_04. Note that p_union_13 and p_union_14 can run simultaneously. P_union_04 can not run with p_union_13 and only starts once p_union_13 has completed.

P_union_14 has only one parameter, library name. It retrieves all records that have a record equivalency flagged Z and builds keyword indexes for those records. After building or rebuilding keywords, p_union_14 updates the record equivalency from Z to N.

P_union_04 also has one parameter, library name. It retrieves all records that have a record equivalency flagged N and builds the equivalency table. Note that equivalencies are symmetric. Therefore, when a record is added to the new record's equivalency table, the new record is in turn added to its equivalency table.

5.3.2 Other processes

In addition to the processes listed above, Union Catalog libraries also have to run p_manage_17 to sort long headings and util_e_08 to build links between the bibliographic library and the authority library. Union Catalog libraries do not have to run util_e_01.

5.4 Deleted Records

In a Union Catalog, record deletion is handled in a distinct way to ensure that records that are intended to be deleted are, and that they are not subsequently overwritten by earlier versions of the record that were loaded out of order. For the purposes of the Union Catalog, a record is considered deleted if it marked as a delete record in the record Leader, that is, if the LDR position 6 is set to d. Union Catalog processes identify the record status and remove all fields from the record except for two, the 005 field and the SID field. (In some versions of the Union Catalog, three fields are retained, 005, SID and LDR.) These fields are retained so that if an earlier or later version of the record is loaded, the load processes can identify the deleted version of the record using the SID field and then compare the transaction dates of the two records to ensure that the correct one is retained. The later record can be a deleted record or a restored record.

5.4.1 Processes that delete records

There are three processes that delete records in the Union Catalog; p_union_08, p_union_20, and p_union_13.

p_union_08

This process should be run after the initial load of records into the Union Catalog. The initial load is usually performed using p_manage_18, p_manage_180 or proc_load_18 depending on the version and the size of the database. (Your project team recommends the appropriate process.) Immediately after loading records, run p_manage_08 to delete records that are status deleted. Note that the process checks the LDR value only; it does not consult any other field in the record. If the record has a Leader status of deleted, then the record is emptied out as described above.

p_union_08 has the following parameters: active library, start record number, end record number, and number of processes.

p_union_20

This service deletes records of a contributing library without disturbing other records and with no need for full re-indexing.

p_union_20 receives the library code (from SID$$b) as a parameter, locates the library's records in the Union Catalog and creates an output file containing the following three fields from each record:

- LDR
- SID
- 005

The 6th position of the LDR field is changed to "d" to indicate that the record should be deleted; field 005 is updated with current date and time.

The output file can then be used as an input file for p_union_13, which deletes the records listed in the file from the database and run the indexing for these records.

The output file produced by p_union_20 is located in the $data_scratch directory, and is named after the contributing library, suffixed by "to_del". For example, "library1_to_del". However, since there might be contributing libraries with a large number of records in the Union Catalog resulting in very large output files and prolonged loading time, it is possible to split the output file by limiting the number of records per file. This is the second parameter of p_union_20. The default value is 10000 records per file; the maximum number allowed is 50000. The output file always has a numbered extension, for example:

library1_to_del.1

library1_to_del.2

library1_to_del.3

p_union_20 can be run either from the Union Catalog Web interface or from the command line. There is no GUI XML interface.

p_union_13
This is the ongoing loader program that loads batches of records into an active Union Catalog. The p_union_13 process checks the Leader status of the record it is loading, and if the status is deleted the record is emptied out, and all but two or three fields are deleted. The fields that are retained are 005, SID and in some versions LDR. In addition to emptying out the record, the p_union_13 process also removes most index access to the record. The SID index is retained.

5.5 Maintenance batch processes

5.5.1 p_union_26

This service transfers Z980 table entries to the Z98 table. Too many Z980 records have a negative effect on the indexing performance.

Please note that union_26 does not run if sent while union_14 is running and that union_13 and union_14 does not run if sent while union_26 is running.

Union_26 can be run from the Union Catalog Web interface or from the command line. There is no GUI XML interface.

6 Appendix: Other Union Catalog Processes

1. p_union_05 – The process p_union_05 creates a report of records that are not themselves considered equivalent; however, they are equivalent

to common records. For example, if system number 25 were equivalent to system number 30 only and system number 30 were equivalent to system number 25 and 35, then 25 and 35 would show up in the report. The process creates a report only; it does not update the database.

Parameters: active library, start record, and end record.

2. p_union_07 – The process p_union_07 can be run on a database after the initial load. It runs through all of the records checking for records with duplicate SIDs. If duplicate SIDs are found, the transaction dates of the duplicate records are checked and the most recent version of the record is retained. The earlier version is purged. The process creates a report of duplicate records and the transaction date of each record; it also updates the database by purging the unwanted records.

   Parameters: active library