Aleph Publishing Mechanism

Version 16 and later

The bridge to knowledge

CONFIDENTIAL INFORMATION

DISCLAIMER

TRADEMARKS

Document released: December 2016

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

Table of Contents

General: Purpose and Scope

The objective of the functionality that is described in this document is to implement a simple mechanism that allows sites to extract records from the ALEPH catalog for publishing purposes (for example, for publishing to search engines and search tools such as Google and Primo).

This publishing platform includes a record extraction of population sets for different ends from a bibliographic or authority library into a repository. The repository is constantly updated. Retrieval of records from the repository can be done to an external system such as Primo.

This document describes the flow and setup needed for successful publishing.

1 Creating and Updating Population Sets via Extract Process

The extract process has two different flows: initial and ongoing. The initial extract usually includes all records in the catalog, while the ongoing extract mainly deals with new and updated records.

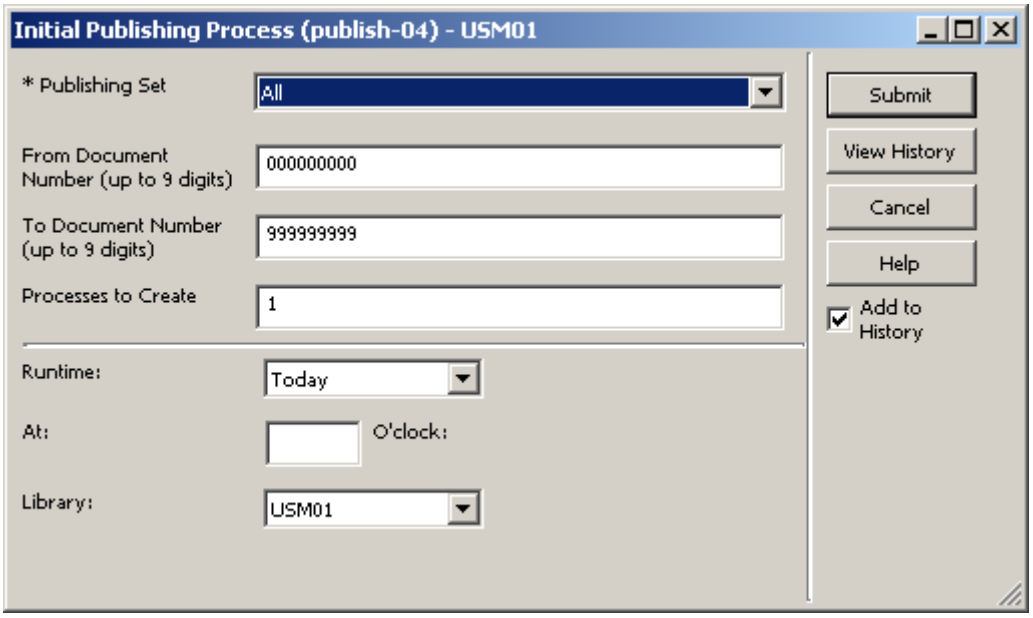
Both publishing processes place the documents into the data repository which is stored in the Z00P Oracle table of the USR library.

Note that the extract process can be performed on the whole database or on specific logical bases. The extract process creates different population sets which are stored in separate Z00P records.

In addition, extracted records can be modified to include information added by standard ALEPH procedures such as FIX and EXPAND.

1.1 Initial Extract Process

The initial extract process is performed by running **Initial Publishing Process (publish-04)**. This service can be run from the **Publishing** submenu of the **Services** menu in the **Cataloging** module.



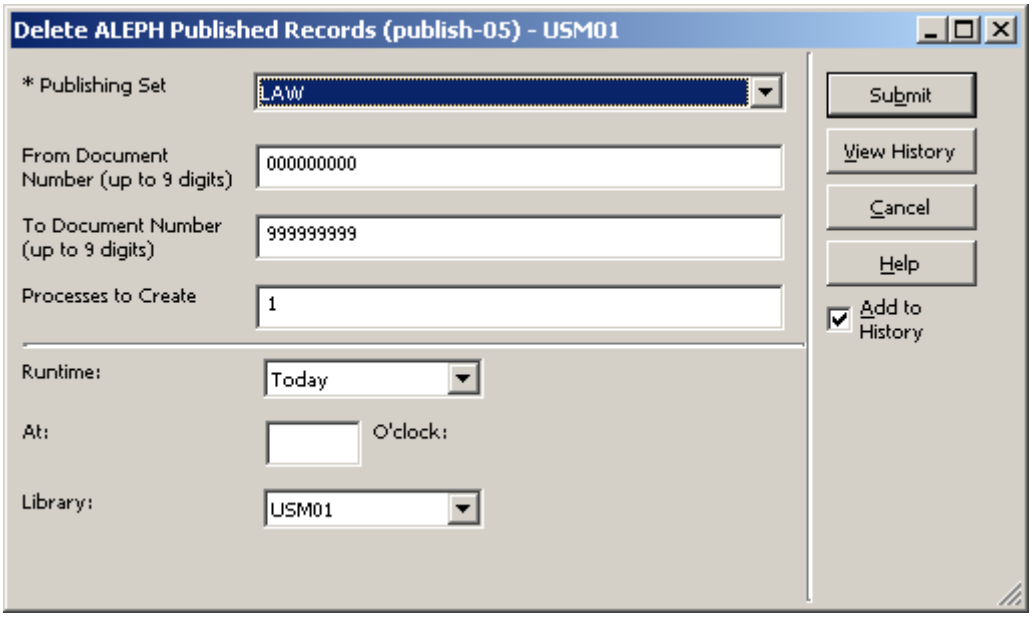
The selected range of records for the specified population set is extracted (ALL is for all sets specified by the System Librarian).

The extraction (initial and ongoing) is performed according to the `tab_publish` table located under the tab directory of the library that contains the records to be extracted (for example, USM01). For more details on how to configure the publishing process, see Section 1.4 *Adding Published Sets*.

If **Initial Publishing Process (publish-04)** tries to upload an invalid xml (containing bibliographic information), a file is written under the `$alephe_scratch` directory. Its name is `publish_04`. The file contains the document number and the library of the document which was not updated due to the invalid xml.

Note: **Initial Publishing Process (publish-04)** works only if the Z00P of the selected population set and its selected record range under the library in which `publish_04` is activated are empty.

In order to delete ALEPH published records that were extracted from the ALEPH catalog for publishing purposes, **Delete ALEPH Published Records (publish-05)** can be used. It has the following parameters: Publishing set, range of records, and number of processes to be created. The selected range of records for a specified set is deleted from Z00P regardless of the original library from which they were extracted.



Note that in order to delete all the sets at once, util/a/17/1 on z00p table should be used.

If a change is made to a base or to a base definition in `tab_base.lng` and this base exists in `tab_publish`, **Delete ALEPH Published Records (publish-05)** and **Initial Publishing Process (publish-04)** should be run to create the initial load again. `UE_21` should be restarted.

1.2 Ongoing Extract Process

The ongoing extract process is required in order to reflect changes to the database such as the deletion of records and updates to the bibliographic records/holdings records/item records, etc. The ongoing extract process has two main stages:

- The trigger for the extract
- The creation/update of repository records

The triggering mechanism for the extract is based on the ALEPH indexing trigger mechanism. In ALEPH, a Z07 record is created for each new or modified record. In the ongoing extract process, when a Z07 record is created for a bibliographic record, the system creates a Z07P record. The Z07P is the trigger for the ongoing extract process.

Note that the creation of z07p is depended on whether `tab_publish` exists in the Bibliographic and/or Authority library. If the table does not exist, no z07p records are created.

Z07 records are created for bibliographic records in various cases such as changes to the related holdings records, authority records, items, etc. This ensures that bibliographic records are indexed not only according to their own data but also according to associated data. Since the Z07P is based on the Z07, it guarantees that

the extracted records, which might contain information derived from FIX and EXPAND procedures, are correctly updated.

Z07p records are also created when an item is loaned, returned, or its hold request status is changed to S (On Hold Shelf).

Z07p is also created for repository records (Z00P) that no longer belong to a set (for example their corresponding bibliographic record has been deleted or its base was changed). The z00p record receives a DELETED status.

The timing of the creation of the trigger record (Z07P) differs depending on whether or not the population set is created based on a logical base. If the population set is not base-dependent, the Z07P is created immediately after the creation of the Z07 record (before it is processed by the `UE_01` indexing daemon). If the population set is base-sensitive, the record must be indexed before it is extracted. In this case, the Z07P record is created only after the Z07 record has been processed by the `UE_01` daemon. The reason for this difference is that in sites where the population sets are not base-sensitive there is no reason to wait for the indexing of the records in order to start the ongoing extract process.

Note that the timing of the creation of the Z07P records explained above is not dependent on the specific population set but depends on whether there is at least one entry in the `tab_publish` table (see Section 1.4 *Adding Published Sets*) that is base dependent. If, for example, there are four population sets defined in the table but only one is base sensitive, then for all sets the Z07P record is created after the processing of the `UE_01` daemon.

1.2.1 UE_21 and UE_22

The handling of the changed Z00P records is performed by the Ongoing Publishing utility, `UE_21`. This utility compares theZ00P record, for which the Z07P was created, with the Z00P record in the repository. If the records differ (after EXPAND and FIX), the Z00P record is handled/changed. When the service finishes processing the triggered documents, the Z07P records are deleted.

The Ongoing Publishing utility, `UE_21`, should be run on a regular basis in order to ensure that the repository is up to date. The Stop Update Publishing Data utility, `UE_22`, stops UE_21.

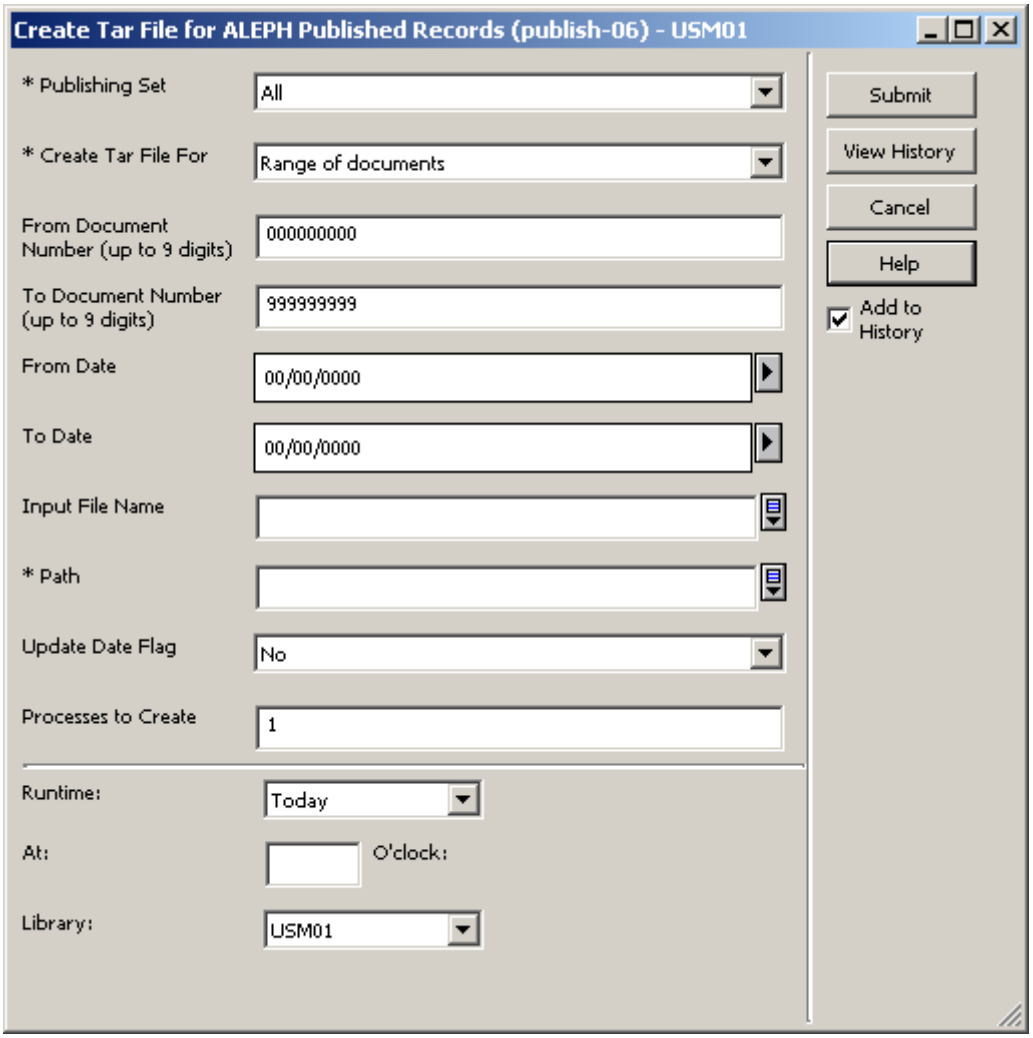
If `UE_21` tries to upload an invalid xml (containing bibliographic information), a file is written under the library's `data_scratch` directory with the following name convention: `util_e_21.xml_err.<YYYYMMDD>.<HHMMSSmm>`. The file contains the document number and the library of the document which was not updated due to the invalid xml. The library should look every couple of days for these files and handle them.

The performance of ue_21 by can be improved by setting the aleph_start/ prof_library variable: `num_ue_21_processes`. This variable enables you to divide the running of the job into several processes. The variable can be set in aleph_start or in the prof_library file of the publishing library. Setting the variable in

$alephe_root/aleph_start or aleph_start.private affects all of the publishing libraries. Setting the variable in $data_root/prof_library of the publishing library affects only this library.

1.3 Retrieval of Repository Records

The changes triggered by Z07P update the Z00P records. **Create Tar File for ALEPH Published Records (publish-06)** can take the updated Z00P records based on dates, record numbers, and an input file and create a tar file for them. This file can be later transferred to different publishing platforms.



If an error occurs while trying to extract a document into the tar file, then that document number is written in an error file. The error file is placed under the library's data_scratch directory.

The name of the file includes the name of the publishing set to which the document belongs and the date and time of the failed extraction. For instance: p_publish_06.err.<Publishing set name>.YYYYMMDDHHMMSSmm. The file includes the document numbers of the failed documents in the format of <Doc Number><Library>, for example 000052114USM01.

1.4 Adding Published Sets

The recommended process of adding new published sets is as follows:

1. Setup the `tab_publish` table (see Section 1.5) with the required definitions of the new set.
2. Run **Initial Publishing Process (publish-04)**. Use the new set name as the parameter, and run the service on all of the documents in the range.
3. Run **Create Tar File for ALEPH Published Records (publish-06)**. Use the new set name as the parameter and run the service on all of the documents in the range.
4. Make sure **Create Tar File for ALEPH Published Records (publish-06)** is set in the `job_list` for execution on your desired frequency includes the new set and is run with the **from last handled date** parameter.

1.5 The tab_publish table: Specifications for the Extraction of ALEPH Records

Both the initial and the ongoing extract process perform the creation and update of records based on the specifications defined in the `tab_publish` table.
Note that the retrieval of repository records does not use this table.

The `tab_publish` table must be located under the tab directory of the library that contains the records to be extracted (in most cases this is the bibliographic and/or the authority libraries).

Following is a sample from the `tab_publish` table:

```
!         1                          2           3   4          5
!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!-!-!!!!!-!!!!!!!!!!!!!!!
TOTAL                                           N ALL    MARC_XML
MEDICINE              MED                       N MED    MARC_XML
LAW                   LAW                       N LAW    MARC_XML
```

Key to the Table:

Column 1 – Publishing Set

This column contains the code of the publishing set of records to be extracted. For example, if the database needs to be extracted in two separate formats for two separate publishing platforms (such as Google and Primo) then two separate sets should be defined in the table. Note that the code must be in upper case.

Column 2 – Base

A set can be the entire database or a section of the database as defined by a logical base. This column contains the code of the desired logical base from the `tab_base.lng` table. If the column is left blank, the entire database is extracted for the set.

Column 3 - De-duplication (for future use)

This column is currently not in use.

Column 4 - Fix and Expand Code

This column contains the fix and expand code of the routines that should be applied before the record is extracted.

Column 5 - Repository Format

This column determines the format of the records in the repository. Currently the supported formats are:

- MARC_XML
- MAB_XML
- OAI_DC_XML (available for ALEPH version 18 and up)
- OAI_MARC21_XML (available for ALEPH version 18 and up)
- HTML (available for ALEPH version 18 and up)

Following is a sample of a record in MAB-XML format:

```
<?xml version="1.0" encoding="utf-8" ?>
- <OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
 - <ListRecords>
  - <record>
   - <header>
      <identifier>aleph-publish:000000006</identifier>
     </header>
   - <metadata>
    - <record xmlns="http://www.ddb.de/professionell/mabxml/mabxml-1.xsd">
       <leader>00242nM2.01200024------u</leader>
       <controlfield tag="LDR">00242nM2.01200024------u</controlfield>
       <controlfield tag="FMT">MU</controlfield>
      - <datafield tag="100" ind1="-" ind2="2">
         <subfield code="a">Bancroft, Hubert H.</subfield>
       </datafield>
      - <datafield tag="300" ind1="-" ind2="2">
         <subfield code="a">Sammlung</subfield>
       </datafield>
      - <datafield tag="331" ind1="-" ind2="1">
         <subfield code="a">The history of California, vol. 4. 1840 - 1845</subfield>
       </datafield>
      - <datafield tag="331" ind1="-" ind2="2">
         <subfield code="a">The works of Hubert Howe Bancroft</subfield>
       </datafield>
      - <datafield tag="410" ind1="-" ind2="2">
         <subfield code="a">San Francisco</subfield>
       </datafield>
     </record>
    </metadata>
   </record>
  </ListRecords>
</OAI-PMH>
```

Note that when a document is removed from a base or deleted, the following status attribute is added to the header tag of the extracted document for the specific base:

```
 - <header>
    <identifier>aleph-publish:000000006</identifier>
   </header>
```

Note that when a document is removed from a base, the following status attribute is added to the header tag of the extracted document for the specific base:

```
- <header status="deleted">
    <identifier>aleph-publish:000000006</identifier>
  </header>
```

2 Expand Programs Related to Publishing

There are 3 new expand programs which serve the publishing process.

2.1 expand_doc_bib_avail

The expand program, `expand_doc_bib_avail`, brings items and holdings availability information. For each HOL record, an AVA line is created with the holding information and its availability.The information is presented in the AVA field which has the following sub fields:

- **$$a: ADM library code**
- **$$b: Sub library code**
- **$$c: Collection** – If there are several items in different collections in one sub library, only one collection of the sub library is presented.
- **$$d: Call Number** – If there are several items in different collections in one sub library, only one call number in the collection is presented.
- **$$e: Availability status** – Can be `available`, `unavailable`, or `check_holdings`. `Available` status is assigned only when there is a certainty that there is an on-shelf item. `Unavailable` is assigned only when there is a certainty that there is no on shelf-item. `check_holdings` is assigned for all other cases (for example, the number of Items is larger than the THRESHOLD, there are holdings but no Items attached, etc.)
- **$$f: Number of Items** (for the entire sub library not just location).
- **$$g: Number of unavailable items** (for the entire sub library not just location).
- **$$h: Multi-volume flag (Y/N)** – If first item's Z30-ENUMERATION-A is not blank or 0 then =Y otherwise = N.
- **$$i: Number of History loans** (for the entire sub library not just location).
- **$$j: Collection code** – If there are several items in different collections in one sub library, only one collection of the sub library is presented.

An item is unavailable if it matches one of the following conditions:

- It is on loan (it has a Z36).
- It is on hold shelf (it has Z37_status=S).
- Items with process statuses are considered unavailable.

Column 3 of `tab_expand` can be used to specify:

- A threshold for checking availability. If the number of items in the sublibrary exceeds the threshold defined in this parameter, then no availability check is performed and the availability is reported as `check_holding`. This value is set by setting column 3 of the `tab_expand` line with:

```
THRESHOLD=xxx.
```
For example:
```
AVAIL  expand_doc_bib_avail     THRESHOLD=050
```

The default threshold is 010.

- The items of item process statuses should be treated as available. This value is set by setting column 3 of the `tab_expand` line with:

  ```
  AVA=xx,yy
  ```

  Where xx and yy are the process statuses that are regarded as available. For example:

  ```
  AVAIL expand_doc_bib_avail     AVA=BD,NW
  ```

- Retrieve availability information by collection (in addition to sublibrary). This value is set by setting column 3 of the tab_expand line with:

  ```
  COLLECTION=Y
  ```

If several parameters are to be defined, a semicolon must separate them, and the threshold must be defined first. For example:

```
AVAIL  expand_doc_bib_avail     THRESHOLD=050;AVA=BD,NW
```

Note:
Information from suppressed HOL records (STA field is SUPPRESSED) is not expanded.
Items that are defined in tab15 as not OPAC displayable (column 10) is not included in the expand.

2.2 expand_doc_bib_avail_hol

The expand program, expand_doc_bib_avail_hol, brings holdings and item availability information, based on the Holding records 852 field and subfields.

The information is presented in the AVA field which has the following sub fields:

$$a ADM library code

$$b Sub library code, based on the 852$$b of the HOL record

$$c Collection text, based on the 852$$c of the HOL record

$$d Call Number - The HOL record's 852 subfields which are set in aleph_start variable: correct_852_subfields (can be 1 or more of the following subfields: hijklm)

$$e Availability status - Can be "available", "unavailable", "check_holdings", or "temporary_location". Available status is assigned if the total number of items minus unavailable items is positive. Unavailable is assigned if the total number of items minus unavailable items is zero or negative. If a record has no linked items (only Holdings records) the status is "check_holdings". If all the items linked to the Holding

records are in a temporary location the status is “temporary_location”. This subfield can be affected by the value of the THRESHOLD parameter in column 3 of tab_expand.

$$f The number of Items that are linked to the HOL record. If no items are linked to the HOL record, it is set to 0.

$$g The number of unavailable items (for the entire sub library not just location) that are linked to the HOL record. If no items are linked to the HOL record, it is set to 0.

$$h Multi-volume flag (Y/N) – If the first item’s Z30-ENUMERATION-A is not blank or 0 then =Y otherwise = N.

$$i The number of loans (for the entire sub library not just location). Based on the HOL record’s linked items. If no item is linked, it is set to 0.

$$j Collection code (852$$c of the HOL record).

$$k Call Number type 1st indicator of 852. If the first indicator of field 852 in the holdings record is 7, the value of 852 subfield $$2 is copied to AVA$$k.

$$p Location priority. A number that represent the priority of the item by its location.

The expand_doc_bib_avail_hol routine consults ./bib_lib/tab/ava_location_priority and AVA$$p is created with a number that represents the location priority.

If there is no match with the ava_location_priority table, no subfield p is created.

If there are 2 HOL records, for example, with the same sublibrary + collection values, then two AVA$$p subfilelds are created with the same priority rank.

$$t Availability text translation. Contains a translation of the content of subfield 'e' (available status), according to the text of the messages entered in table ./errror_lng/expand_doc_bib_avail (the same one as used by expand_doc_bib_avail).

$$7 Holdings ID - Contains the HOL library code and the HOL record number (e.g. USM60000000741) for which the AVA is created. Non-relevant for AVA fields of items with no linked HOL record.

Routine Additional Parameters

To define additional subfields added to the AVA fields from the 852 field of the HOL, set the parameter `SF` in column 3 of tab_expand (for example: `SF=z,t`) to copy subfields $$z and $$t. Note that the additional subfields that are copied from the HOL record 852 to the AVA field override the subfields created this expand program.

To define the maximum number of items to check per sublibrary, set the following parameter in column 3 of tab_expand: `THRESHOLD=080`. In this example, the maximum number of items per sublibrary is 80. Note that the number set in this parameter must have three digits. This parameter affects the content of subfield $$e (availability status) and sub field $$t (translation of $$e).

Items with process statuses are considered “unavailable”. Note that Col.3 of tab_expand can be used to specify item process statuses that their items should be treated as “available”.

To ignore item process statuses that should be treated as “available”, set the following parameter in column 3 of tab_expand: `AVA=BD,MK`. It is possible to specify more than one process statuses, delimited by a comma “,”.

If you want to take reshelving time into account; set the following parameter in column 3 of tab_expand: `RESHELVING=Y`. This is mostly relevant for Real Time Availability functionality (availability X service).

For example:

```
!   1                   2                              3
!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
FULLP       expand_doc_bib_avail_hol        THRESHOLD=050;AVA=BD,NW;SF=z,t
```

In this example:
The THRESHOLD parameter limits the number of items per sublibrary to 50.
The AVA parameter defines items with process status BD or NW as available.
The SF parameter adds 852$$z and 852$$t to the AVA$$z and AVA$$t. The SF parameter supports multiple subfield occurrences.

Note:

For items that are not related to any HOL records, an AVA field is created for each sublibrary and collection combination similar to expand_doc_bib_avail, as if `COLLECTION=Y` is defined and without consulting the `SF` parameter.

2.3 expand_doc_del_fields

This expand program deletes all the fields in the record except the fields specified in Col.3 of `tab_expand`. The fields in Col.3 are separated by a comma.

Example:

```
!   1                   2                              3
!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
TEST        expand_doc_del_fields           245##,260##,500##,AVA##
```

Only 245##, 260##, 500## and AVA## fields are retained.

2.4 expand_doc_bib_accref_1

This expand program works like `expand_doc_bib_accref`. The difference is that the cross reference information expanded by `expand_doc_bib_accref_1` is put into lines named after the acc code of the relevant Z01 record of the bibliographic library.

If, for example, the acc code of the relevant Z01 record is AUT, then the line to which the information is expanded is called AUT.

3 Notes on Implementing the Publishing Mechanism for Primo

3.1 General

For Primo, the publishing mechanism should be setup in the Bibliographic library.

In terms of expands, you should include the new expands for availability (`expand_doc_bib_avail`, `expand_doc_bib_avail_hol`) and cross-references

(expand_doc_bib_accref_1.) Do not use expand_doc_bib_accref because Primo needs to be able to distinguish between preferred and non-preferred terms. In addition, if you need fields from the holdings records, e.g. 856 fields, include an expand that adds the HOL record to the BIB. (However, the LDR and 008 fields of the HOL must be removed. See below an explanation how to do this).

If you plan to implement both a regular pipe and an availability pipe you should use the following hardcoded sets:

- **PRIMO-FULL** – Should include the bibliographic record and all related expanded data (authority, holdings and availability)

- **PRIMO-AVAIL** – Should include only availability information

Note that the disk space you require is 12KB per record for the PRIMO-FULL set and 9KB per record for the PRIMO-AVAIL set.

Tab_publish

Here is an example for the tab_publish setup:

```
!        1                       2           3   4          5
!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!-!-!!!!!-!!!!!!!!!!!!!!!
PRIMO-FULL           PRM01_PAC             N PRM1  MARC_XML
PRIMO-AVAIL          PRM01_PAC             N PRM2  MARC_XML
```

Expands

The following expands should be added to tab_expand for PRIMO-FULL

```
expand_doc_bib_avail
expand_doc_bib_accref_1
expand_doc_bib_hol
```

In addition, if you need the HOL record, use an expand like expand_doc_bib_hol The complete HOL is added, but the control fields should be removed. (007 is retained as it can contain useful information and has the same format as that of the BIB). This can be done by using Column 3 in the tab_expand table as shown in the example below.

The following expands should be added for PRIMO-AVAIL

```
expand_doc_bib_avail
expand_doc_del_fields
```
– To delete all fields except for the availability field.

Example:

tab_expand:

```
PRM1        expand_doc_bib_avail
PRM1        expand_doc_bib_accref_1
PRM1        expand_doc_bib_hol          -001,002,003,004,005,006,008

!
```

```
PRM2         expand_doc_bib_avail
PRM2         expand_doc_del_fields          AVA##
```

Tab_fix

The following setup is recommended to prevent the updating of Z00P due to a change in the 005 field (which is updated whenever the BIB record is saved in the cataloging client):

```
! 1                  2                                  3
!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
PRM1  fix_doc_do_file_08             del_005
```

The parameter file (del_005) should be located in the library's import directory under the tab directory.

```
!   2   3  4  5   6   7                 8                            9
!-!!!!!-!!-!-!!!-!!!-!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!
1 005                          DELETE-FIELD
```

3.2 Setup for MAB Records Extract

The following is a setup example for publishing a MAB library.

3.2.1 $data_tab/tab_publish

```
!        1                        2           3   4          5
!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!-!-!!!!!-!!!!!!!!!!!!!!!
PRIMO-FULL                                      PRIMO MAB_XML
PRIMO-ONGOING                                   PRIMO MAB_XML
```

3.2.2 $data_tab/tab_expand

```
!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
PRIMO      expand_doc_sysno
PRIMO      expand_doc_mab_recursive       EXPAND-SON=PM-FAMILY,EXPAND-FATHER=PM-FAMILY,FIXDOC-SON=,FIXDOC-FATHER=
PRIMO      expand_doc_bib_avail

PM-FAMILY  expand_doc_mab_aut_ref         100##,4,196##,I-SF=9a,I-CODE=IDN,FIX-AUT=PMREF,PREF=100##,CROSS=101##,CROSS-DOC=101,TAG-MAX=99999
PM-FAMILY  expand_doc_mab_aut_ref         800##,6,824##,I-SF=9a,I-CODE=IDN,FIX-AUT=PMREF,PREF=800##,CROSS=801##,CROSS-DOC=801,TAG-MAX=99999
PM-FAMILY  expand_doc_mab_aut_ref         200##,4,296##,I-SF=9a,I-CODE=IDN,FIX-AUT=PMREF,PREF=200##,CROSS=201##,CROSS-DOC=201,TAG-MAX=99999
PM-FAMILY  expand_doc_mab_aut_ref         802##,6,826##,I-SF=9a,I-CODE=IDN,FIX-AUT=PMREF,PREF=802##,CROSS=803##,CROSS-DOC=803,RAG_MAX=99999
PM-FAMILY  expand_doc_mab_aut_ref         700##, ,     ,I-SF=aa,I-CODE=NNS,FIX-AUT=PMREF,PREF=700##,CROSS=701##,CROSS-DOC=701,TAG-MAX=99999
PM-FAMILY  expand_doc_mab_aut_ref         902##,5,947##,I-SF=9a,I-CODE=IDN,FIX-AUT=PMREF,PREF=902##,CROSS=952##,CROSS-DOC=952,TAG-MAX=99999
```

Expand_doc_mab_recursive is called up from $data_tab/tab_expand (BIB library) by using the expand menu “PRIMO”. The new expand can be configured in the following way (tab_expand, col. 3), each entry has to be separated by a comma:

EXPAND-SON= Expand menu which should be used for expanding information from the current record (=son)
EXPAND-FATHER= Expand menu which should be used for expanding information from the father record into the son
FIXDOC-SON= Fix routine which should be used for the current record (=son)
FIXDOC-FATHER= Fix routine which should be used for the father record

Expand_doc_mab_aut_ref takes preferred term and cross-references for authorities from the authority record into the BIB record. This includes classifications (700) and subjects (9xx). Expand_doc_mab_aut_ref can be configured in the following way (tab_expand, col. 3). Each entry is separated by comma:

First source field BIB record which contains preferred term,
Increment value (optional),
Last source field BIB record which contains preferred term (optional),
I-SF = <Subfield BIB record which should be used to identify the corresponding AUT record><Subfield AUT record which contain the preferred term>,
I-CODE= Direct index which should be used to identify the AUT record,
FIX-AUT= Name of the fix routine which is used to transform the relevant fields within the AUT records into a format that is similar to the BIB fields (tab_fix, col.1, special description see below)
PREF= Field in the AUT record which contains the preferred term after the fix procedure called by FIX-AUT
CROSS= Field in the AUT record which contains the cross references after the fix procedure called by FIX-AUT
CROSS-DOC= Destination field in the BIB record for cross references
TAG-MAX= Max. no. of preferred terms and cross references per authority record to take over into the BIB record.

The fix routines that are called from “expand_doc_mab_aut_ref” (FIX-AUT=) have to be defined in ALEPH table $data_tab/tab_fix (AUT library). The fix routine is needed to transform the relevant fields (preferred term and cross references) within the AUT record into a format that is similar to the BIB fields. This is described below.

Col. 3 contains program arguments; in this case “tab_fix_convit_ref_pm” is a configuration file which contains the definitions about the source fields and destination fields (see below).

3.2.3 $data_tab/tab_fix and corresponding tab_fix_convit_ref_pm

```
! 1                 2                                3
!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
PMREF fix_doc_convit                FILE=tab_fix_convit_ref_pm
```

The table "tab_fix_convit_ref_pm" must be added to authority files. The table tab_fix_convit_ref_pm must exist in each authority library, in the tab directory.

The following entries should be taken by default:

./lib10/tab/tab_fix_convit_ref_pm

```
! 1   2   3   4               5
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
800   a 100   a
800a  a 100   a

800b  a 101   a
800c  a 101   a
820#  a 101   a edit_field   SEL-I=-:v:
830_  a 101   a
```

./lib11/tab/tab_fix_convit_ref_pm

```
! 1   2   3   4               5
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
800_  a 200   a

801b  a 201   a
803_  a 201   a
810_  a 201   a edit_field   SEL-I=-:v:
811b  a 201   a
812_  a 201   a
813b  a 201   a
814_  a 201   a
815b  a 201   a
816_  a 201   a
817b  a 201   a
818_  a 201   a
819b  a 201   a
820_  a 201   a
821b  a 201   a
822_  a 201   a
823b  a 201   a
824_  a 201   a
825b  a 201   a
826_  a 201   a
827b  a 201   a
828_  a 201   a
829b  a 201   a
830_  a 201   a
831b  a 201   a
832_  a 201   a
833b  a 201   a
834_  a 201   a
835b  a 201   a
836_  a 201   a
837b  a 201   a
838_  a 201   a
839b  a 201   a
840_  a 201   a
841b  a 201   a
842_  a 201   a
843b  a 201   a
844_  a 201   a
845b  a 201   a
846_  a 201   a
847b  a 201   a
848_  a 201   a
849b  s 201   a
```

./lib12/tab/tab_fix_convit_ref_pm

```
! 1   2   3   4               5
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
800_  a 902   a
801_  a 902   a
802_  a 902   a
803_  a 902   a
804_  a 902   a
805_  a 902   a
820_  a 952   a
821_  a 952   a
822_  a 952   a
823_  a 952   a
824_  a 952   a
825_  a 952   a
830_  a 952   a
```

./lib13/tab/tab_fix_convit_ref_pm

```
! 1   2   3   4               5
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
800_  a 700   a

830_  a 701   a
```

4 Technical Information for the ALEPH Publishing Mechanism

4.1 Information Required for the Maintenance of the Z00P Oracle Table

4.1.1 Disk Space, Definitions, and Basic Concepts

Entity – A representation of an ALEPH record in the Publishing platform in a certain format and in a certain type of content. Entities are stored in Oracle as a part of the ALEPH database.

Basic Entity – An entity that is in XML format containing data from the basic bibliographic record (with no expands).

Disk Space required:

	Basic Entity	**Primo Entity - Full**	**Primo Entity - Avail**
Oracle Repository	12K per entity	16K per entity	2K per entity
Oracle Repository + Tar	13K per entity	17K per entity	3K per entity

4.1.2 Oracle Recommendations

- When running **Initial Publishing Process (publish-04)** there are two possibilities:

- Turn off Oracle Archiving before starting to run the job. After the job is finished, Oracle Archiving should be turned on and the database should be fully backed up (hot or cold backup).
  The implications of choosing this possibility are that data integrity is not kept during the time Oracle Archiving is off. If the data needs to be recovered, it can only be brought back to the point of time before the job has started. The benefits of using this mode is better performance and less disk space.

- Keep Oracle archiving on. The disk space allocated for Z00P should be doubled from the sizes given above. This means the space available for the archive files should be doubled, not the Oracle tablespace.
  The implication of choosing this possibility is a slower throughput in the process (less entities/sec). The benefit is keeping data integrity at all times.

- It is recommended that the publishing entity (Z00P) Oracle table have a separate tablespace to help with maintenance
- To increase the throughput , the following setup should be applied:
  - Enlarge the redo file from 50MB to 500MB. This should be done by an Oracle DBA.

  - It is recommended that the redo files be on a different physical disk than the tablespace.

  - If possible, the new tablespace should be on a different physical disk than the already existing ALEPH tablespace.

4.2 Initial Publishing Process (publish-04) Parallelism Recommendations

The Initial Publishing Process (publish-04) is a one time process. Therefore, it is recommended to run this process when the library's activity is minimal, such as during weekends and holidays. In order to maximize the throughput of the process and minimize the run time, it is recommended to use a large number of parallel processes. As a rule of thumb, it is recommended to use 70%-80% of the logical CPUs of the ALEPH server.

For setting a better fine-tuned configuration, it is recommended to monitor the utilization of the server and decide accordingly.

4.3 Implementation Notes

4.3.1 Z00P/Z07P Creation

- If the `file_list` templates under `./aleph/tab/` are not being used, add the following lines to the `file_list` under the BIB/AUT libraries as in the following sample from the demo libraries.
  Note that the Z07P table should be created in the same tablespace as existing ALEPH tables, and the Z07P indexes should be created in the same tablespace as existing ALEPH indexes.

```
!1      2           3          4          5      6
!       !           !          !          !      !
TAB z07p            100K       100K       ts0
IND z07p_id         100K       100K       ts1
```

- In the BIB/AUT libraries, create the Z07P table using `Util A / 17 / 1`

- The publishing library is by default the usr_library. However, it is possible to use a different publishing library by setting the 'publishing_library' environment variable in the aleph_start file.

- If the `file_list` templates under `./aleph/tab/` are not being used, or if using a library other than the usr_library is required for storing the published documents, add the following lines to the `file_list` under the usr_library (or the library that has been set as the 'publishing_library'), as in the following sample from the demo libraries.

  Note that the sizes given are for demonstration purposes only. For actual sizes, see Section 4.1.1 Disk Space, Definitions, and Basic Concepts.

```
!1      2           3          4          5      6
!       !           !          !          !      !
TAB z00p            2M         1M         tslob
IND z00p_id         100K       100K       ts1
IND z00p_id1        100K       100K       ts1
IND z00p_id2        100K       100K       ts1
IND z00p_id3        100K       100K       ts1
IND z00p_id4        100K       100K       ts1
SEQ last_publish_seq
```

- In the usr_library (or the library that has been set as the 'publishing_library'), create the Z00P table using `Util / A / 17 / 1`

In the usr_library (or the library that has been set as the 'publishing_library'), create the Oracle sequence `last_publish_seq` using `Util / A / 17 / 9 / 2`

4.3.2 Configuration Table

- Create the table `tab_publish` under $data_tab of the **BIB/AUT** library

- Use `Util / H /` 2 (under the relevant BIB/AUT library) to synchronize the header of `./aleph/headers/libnn/tab/tab_publish`

4.3.3 JAVA Environment Configuration

- Add the following lines to `./alephe/aleph_start`

  - Before the line:

```
switch ($platform_type)
```

add line:

```
setenv JAVA_MACHINE `uname -p`
```

- For linux machines – in $platform_type switch, at the end of case 5, after the line:

  ```
  setenv   JAVA_HOME
  ```

  add line:

  setenv JAVA_MACHINE i386

- Add the lines:

  ```
  setenv    JAVA_ENABLED   TRUE
  setenv LD_LIBRARY_PATH
  "${LD_LIBRARY_PATH}:$JAVA_HOME/jre/lib/$JAVA_MACHINE"
  setenv LD_LIBRARY_PATH
  "${LD_LIBRARY_PATH}:$JAVA_HOME/jre/lib/$JAVA_MACHINE/serv
  er"
  ```

- AIX users -
  - Make sure that Java 1.4.2 or higher is installed. On AIX machines, Java is under $ORACLE_HOME/jdk. To install the required Java version, refer to the Aleph Installation Guide.

  - In your $alephe_root/aleph_start file:

    1. In the definition of the environment variable LIBPATH (under the "case AIX:" section), make sure that "${JAVA_ HOME}/jre/bin/classic" and "${JAVA_HOME}/jre/bin" are BEFORE the system library "/usr/lib". For example:

    ```
    setenv   LIBPATH
    "$aleph_product/lib:$aleph_product/local/libxml/lib:$
    {JAVA_HOME}/jre/bin/classic:${JAVA_
    HOME}/jre/bin:/usr/lib:${LD_LIBRARY_PATH}"
    ```

    2. Add the following line immediately after the definition of LIBPATH (under the "case AIX:" section), and before the "breaksw" statement:

    ```
    setenv LDR_CNTRL USERREGS
    ```

- Shutdown Aleph and restart it using:

  ```
  aleph_shutdown
  aleph_startup
  ```

4.3.4 Daemon Handling

- Restart `ue_01` daemon under relevant BIB/AUT libraries.

- In order to activate the new daemon for the ongoing update of the publishing platform, under relevant BIB/AUT libraries, use:

```
Util / E / 21
```

In order to stop the new daemon use:

```
Util / E / 22
```

4.3.5 Availability Updates

In order to publish records linked through an ITM link when item availability is changed (loan, return, etc.) in adm_library (for example, USM50), add to [adm_library]/tab/tab_z105 the following line:

UPDATE-ITM m USM50

Restart ue_11 in $z105_library

4.3.6 Additional Notes

- For ALEPH v. 21.1.3 and later:
  It is possible to split the tar file created by the **Create Tar File for ALEPH Published Records (publish-06)** batch service into several files. This can be done by setting a "block size" in the $data_root/prof_library of the BIB library as follows:

  ```
  setenv p_publish_06_block  5000
  ```

  This definition sets a limit on the number of records in each output file.
  The default is 50000.

- For ALEPH v. 21.1.4 and later:
  It is possible to keep the old log files created by the Create Tar File for Aleph Published Records (publish-06) batch service. This can be done by setting the "keep_log_files" parameter to Y in the $data_root/prof_library of the BIB library as follows:

  ```
  setenv keep_log_files Y
  ```

- For ALEPH v. 22.1.0 and later:
  It is possible to keep the empty directories created by the Create Tar File for Aleph Published Records (publish-06) batch service.
  This can be done by setting the "keep_publish_dir" parameter to Y in the $data_root/prof_library of the BIB library as follows:

  ```
  setenv keep_publish_dir Y
  ```

- For ALEPH v. 16.02 on a Solaris machine, a GNU tar should be used.
  The following command should be used:

  ```
  cp $alephm_proc/gnu_tar $aleph_product/bin/tar
  source $alephe_root/aleph_start
  ```

- For v. 16.02 only:

1. Replace the old version of `./alephe/error_eng/p_publish_04` with the new one.

2. Add the files:

```
./alephe/error_eng/ue_21
./alephe/error_eng/p_publish_05
./alephe/error_eng/p_publish_06
./alephe/pc_b_eng/p-publish-04.xml
./alephe/pc_b_eng/p-publish-05.xml
./alephe/pc_b_eng/p-publish-06.xml
```

3. In the file `./alephe/pc_b_eng/menu-catalog.xml`
   - Remove the lines:

```
   <item>
            <display>Ongoing Publishing Process
(publish-03)</display>
            <file>p-publish-03</file>
      </item>
```

   - Add the lines:

```
   <item>
              <display>Delete ALEPH Published
        Records (publish-05)</display>
          <file>p-publish-05</file>
     </item>
     <item>
              <display>Create Tar File for ALEPH
        Published Records   (publish-06)</display>
         <file>p-publish-06</file>
     </item>
```

   - Add the files:

```
./alephe/error_eng/ue_21
./alephe/error_eng/p_publish_06
./alephe/pc_b_eng/p-publish-05.xml
./alephe/pc_b_eng/p-publish-06.xml
```