VERSION 3

DigiTool to Aleph Metadata Synchronization

Last Update: June 2012
Document Version 3.0
Code: D-ver3-DTL2ALEPHSync-1.1

Documentation produced November 2010

Document version 3.0

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

TABLE OF CONTENTS

1 Scope and Purpose

The DigiTool metadata synchronization service provides a way to synchronize certain types of metadata from DigiTool to Aleph and from Aleph to DigiTool. The Aleph catalog is often the master catalog for object-related metadata. The matching is done by an identifier within the metadata, for example, in MARC the control field 001. The content of this field is matched between DigiTool and Aleph records. Using OAI harvesting/providing, DigiTool sends new metadata records to Aleph to be maintained and cataloged further. DigiTool keeps existing Aleph records updated by providing persistent links back to the delivery system for objects that share the metadata in DigiTool.

Data is passed back and forth from both systems by leveraging the OAI-PMH capabilities of both systems.

2 About the workflows

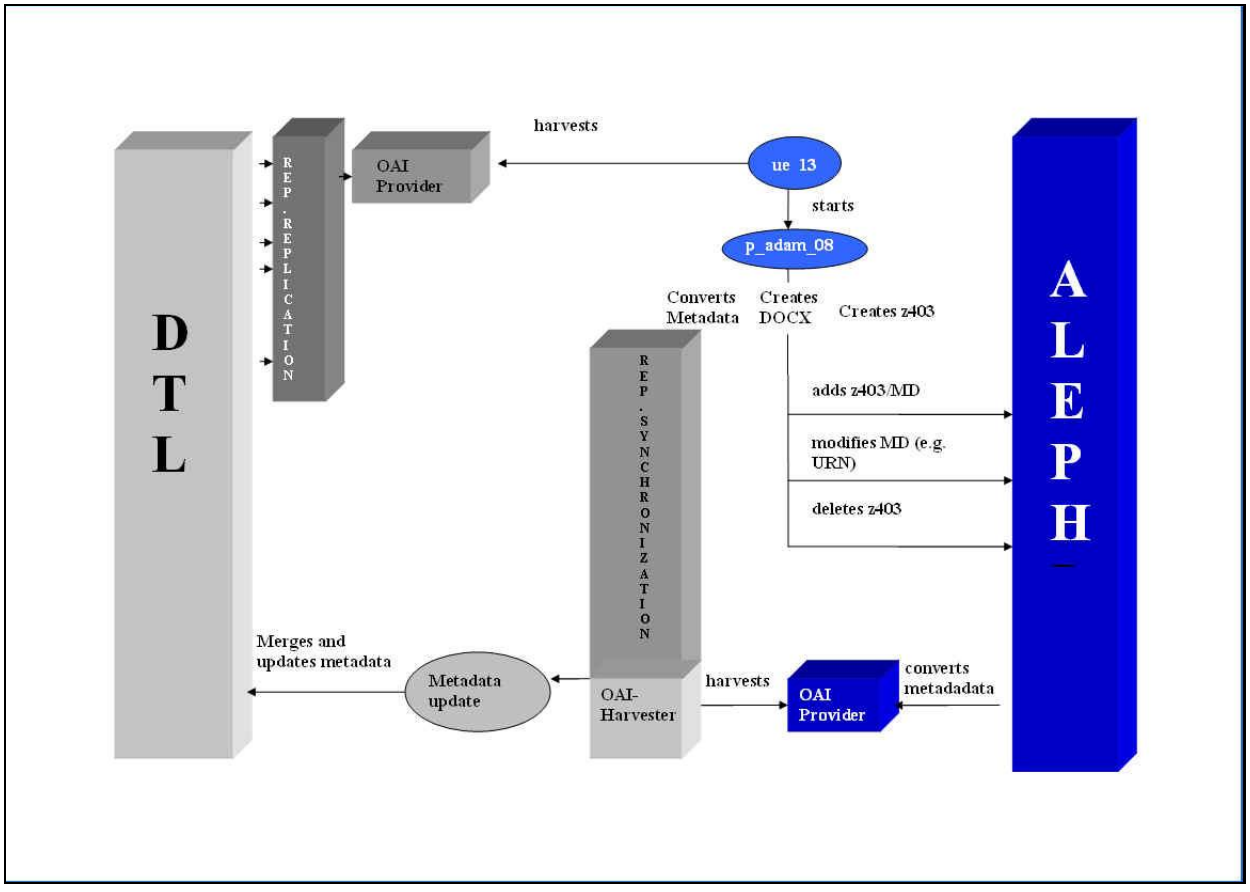


Figure 1: DigiTool - Aleph Synchronization

Two typical workflows of the DigiTool-Aleph synchronization are as follows:

Scenario #1

1. An object (for example, a book) is cataloged in Aleph, scanned, and digitized.
2. The scanned object(s) (that is, file streams) are ingested into the DigiTool repository. During the ingest process, the object gets a set of "basic" descriptive metadata, including a pre-defined field wherein the identifier of the Aleph metadata is provided (for example, the 001 field). The content of the field is used for matching between the data on the Aleph and DigiTool sides.
3. Once the objects have been loaded into DigiTool, a persistent link and PID are associated with this file stream. Aleph needs to automatically record this delivery link to make it accessible from the Aleph record/OPAC.
4. In order to achieve this, DigiTool defines a repository replication set which allows writing out digital entities (records) from the DTL repository to the OAI_PUB database table in DigiTool. The setup gives the opportunity to define which set or population should be replicated and ultimately provided to Aleph.
5. The replicated set of digital entities is then defined as an accessible OAI set in the relevant configuration table `oaipubconf.xml` under the `dtle/tab/oai` directory for providing to Aleph.
6. Now the provided data can be harvested by Aleph using the service ue_13. The data is taken to the Aleph system for update/addition/deletion of matching 001 field records.
7. The `ue_13` (a background daemon) itself brings the data in to Aleph, then starts the `p_adam_08` service which converts the incoming metadata and matches against Aleph library bibliographic records. In our scenario, the matching 001 field is our key. This key allows the persistent delivery link to populate the correct Aleph record(s). It is stored in the Z403 table of the Aleph library.
8. Since Aleph contains the master catalog, DigiTool will need to benefit from the full and most recent updates to the Aleph record as it continues to be cataloged. In order to achieve this, DigiTool—this time the harvester—harvests the master cataloged metadata from Aleph's—now the OAI provider—available and provided OAI data set.
9. This data is matched against the unique 001 identifier in DigiTool—and the Aleph record merges/overrides the analogous DigiTool one—ensuring a synchronized record in both systems.
10. The ongoing synchronization relies on periodic checks between DigiTool and Aleph to see what changes, additions, and/or deletions have been made in both systems. The OAI provider of both systems requests data by date range, and the provided data within any date range is based on the last updated date of records in both systems.

Scenario #2

1. Basic metadata is submitted with a digitized file stream(s) and is deposited/ingested into DigiTool. For instance, a professor deposits a born-digital thesis or e-Journal article and submits the basic metadata with file stream through

the deposit workflow with some basic Dublin Core metadata—filled in using the pre-defined deposit metadata form.

2. The record may need further cataloging and should ultimately be stored in the master Aleph catalog, though it does not yet exist there.
3. DigiTool replicates and makes the data OAI-accessible to Aleph.
4. Aleph OAI harvester (ue_13) harvests the OAI data from DigiTool
5. Based on a field INS with content INSERT, Aleph knows that it must add a new system number or bibliographic record with the basic metadata provided by DigiTool. In this case, we may use a local field—that is, dc:ins with content = INSERT—and map this field, once in Aleph, to a MARC INS field, which Aleph recognizes. In other words, the metadata is provided in Dublin Core, and Aleph converts it to the Aleph-native metadata format—usually MARC. This is performed through an Aleph library-specific fix_routine. Additionally, while adding the new Aleph record and "converted" MARC data, a persistent link is provided which allows access from this new Aleph record to the DigiTool delivery system in order to view the digital file stream from the Aleph OPAC.
6. Cataloging ensues on the Aleph side and the MARC record is now full and complete.
7. DigiTool synchronization ensues: Aleph OAI provides the updated master cataloged record back to DigiTool in the DigiTool stored format (in our case, DC, the reverse of the above fix_routine) and merges/overrides the existing DigiTool metadata record, ensuring a synchronized metadata record in both systems.

3 DigiTool to Aleph setup

The synchronization direction by which DigiTool provides data to Aleph requires setup on the DigiTool side—that is, what to provide and how to provide it—and on the Aleph side—that is, what to request from DigiTool and how.

3.1 DigiTool

3.1.1 Repository replication

DigiTool's Repository Replication provides a way to replicate items from the repository to the database table named OAI_PUB. In our case, Aleph expects a certain format of replicated data in order to match and update data properly. The format is based on an XSL stylesheet, `de2aleph.xsl`, which takes DigiTool records or digital entities in a given set population and strips all unnecessary information so that only basic control data, links, and descriptive metadata are retained—the data Aleph can use.

For more information on creating repository replications, refer to the DigiTool General Configuration Guide, Repository Replication and OAI section.

For our example, we will provide a sample Aleph-ready configuration for repository replication.

```
<replication name="dtlaleph" enable="true">
             <rs:set>
              <control_fields>
                  <logical_operator logical_op="and">
                          <field key="owner"
op="exact">DTL01</field>
                      <field key="usagetype"
op="exact">VIEW</field>
                  </logical_operator>
              </control_fields>
              <require_md name="descriptive" type="dc"/>
              <require_md name="descriptive" type="marc"/>
              <parent_only>true</parent_only>

<manifestation_leader_only>true</manifestation_leader_only>
          </rs:set>
<targets>
  <target type="class"
class_name="com.exlibris.digitool.repository.de.DigitalEntityO
AIReplicator">
    <params>
      <param name="db_url">dbc:oracle:thin:@myserver.
de:1521:dtl3</param>
      <param name="db_username">d31_oai01</param>
      <param name="db_password">d31_oai01</param>
      <param name="db_table_name">oai_pub</param>
      <param name="set_spec">dtlaleph</param>
      <param name="metadata_format">de2aleph</param>
    </params>
  <format_convertor type="xsl" convertor="de2aleph.xsl"/>
  </target>
</targets>
          <scheduling sync_on_startup="true"
type="every_x_hours" x="24"/>
      </replication>
```

3.1.2 Repository OAI accessibility

The appropriate data set is replicated to database table OAI_PUB and now needs to be made OAI-accessible, allowing the synchronization service to pass data back and forth by way of the OAI-PMH protocol (HTTP).

The following is an example of how this can be achieved:

```
<?xml version="1.0" encoding="UTF-8"?>
<oairoot>
<set>
<setSpec>oai</setSpec>
<setName>oai set</setName>
<setDescription>The default oai set</setDescription>
</set>
<set>
<setSpec>history</setSpec>
<setName>History</setName>
</set>
```

```
<set>
<setSpec>dtl2aleph</setSpec>
<setName>dtl2aleph</setName>
</set>
<repositoryName>ExLibris Repository</repositoryName>
<adminEmail>Me@exlibris.co.il</adminEmail>
<description>
<oai-identifier
xmlns="http://www.openarchives.org/OAI/2.0/oai-identifier"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oaiide
ntifier
http://www.openarchives.org/OAI/2.0/oai-identifier.xsd">
<scheme>oai</scheme>
<delimiter>:</delimiter>
<sampleIdentifier>oai:il-dtldev02:52719</sampleIdentifier>
</oai-identifier>
</description>
</oairoot>
```

Once the OAI data is indexed, the data should be accessible to any OAI requester/harvester—specifically Aleph's.

3.2 Aleph

3.2.1 Getting Started

Perform the following steps to get started synchronizing Aleph with DigiTool.

1. Aleph license update—contact your local EXL representative
2. Aleph v 16 only—Add z403 table to the library or libraries needing synchronization:
   a) Add the following lines to `$data_root/file_list`:

```
TAB z403          4M          0K          TS2D
IND z403_id       4M          0K          TS2X
IND z403_id1      4M          0K          TS2X
```

   b) UTIL A 17 1 to create z403, z403_id and z403_id1

3. Add table `tab_ue13.conf` to `$data_tab`—not in version by default. (And any tab_fix tables needed to map metadata—optional—and not in version by default.) See sections below for more details.

3.2.2 $data_tab/tab_ue13.conf (configure Aleph OAI harvester)

The table is used to configure the Aleph OAI harvester (`ue_13`). The following parameters have to be defined for `ue_13`—located in the Aleph library's tab unit:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[Main]
BASE-URL = http://server:port/oai/provider
!Base URL of the (DigiTool) OAI provider.

DELETE-XML-FILES = N
!Options (Y/N) - Delete XML files from $alephe_scratch after
synchronizing the records.

DELETE-LOG-FILES = N
!Options (Y/N) - Delete log files from $alephe_scratch after
synchronizing the records.

METADATA-PREFIX = de
!Format which has been provided from DigiTool, for example, "de" for
digital entity format. The value is based on the defined format of
the set defined in the DigiTool OAI provider.

RECALL-PERIOD = 24h
!Value for the time period for every harvesting step (in minutes
unless otherwise specified – h = hours). It is a time range for each
OAI request – similar to the "Scope" parameter on the DigiTool side.
This parameter can contain two digits. This means that the maximum
amount of minutes possible is 99. For values larger than 99 minutes -
specify "h" (hours).

SERVICE-NAME = oairep.pl
!Perl script which is used by ue_13.

START-DATE = 2006-01-01-T00:00:00Z
!Time stamp (in UTC) for provided data. The records must have been
provided after this date.
!Syntax: YYYY-MM-DD-THH:MM:SSZ This parameter is only relevant for
the first run of the ue_13 and from then on will be run according to
the ue-13 last date and time.

TIMEOUT = 900
!Timeout value for ue_13 OAI harvester (in seconds).

WAIT-TIME = 216000
!Waiting time before checking for new records (in seconds). For
instance, 216000 seconds = 24 hours.

[Debug]
VERBOSE-MSG = 2
!The amount of information to be printed in the log files (1, 2)

[Sets]
DTLALEPH1 = dtlaleph
!Name of the DigiTool OAI set.

DTLALEPH1-RUN-01 = csh -f $aleph_proc/p_adam_08
USM01,<filename>,<setname>,,Y,Y,001,dc:DCUSM%marc:USUSM,

!Command to start service adam-08.
```

3.2.3 $data_tab/tab_dc (DC to MARC conversion)

The `tab_dc` table is used for Dublin Core to MARC conversion—that is, when data supplied from DigiTool is in Dublin Core and needs to be mapped into MARC in Aleph.

The conversion of DC elements to Aleph categories is done in two steps. First, the DC metadata are converted to Aleph sequential format. Second, Aleph sequential format is converted to MARC 21 (or other Aleph metadata—that is, MAB, UNIMARC, and so on).

Structure of the table:

```
Col. 1:   Internal field code
Col. 2:   Dublin Core Element

!1              2
!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
001   dc:title
002   dc:creator
003   dc:subject
004   dc:description
005   dc:publisher
006   dc:contributor
007   dc:date
008   dc:identifier
009   dc:source
010   dc:language
011   dc:relation
012   dc:coverage
013   dc:rights
014   dc:type
015   dc:format
016   dc:ins
017   dc:alephsync
020   dcterms:alternative
021   dcterms:abstract
022   dcterms:tableOfContents
023   dcterms:spatial
024   dcterms:temporal
025   dcterms:extent
026   dcterms:medium
027   dcterms:available
028   dcterms:issued
029   dcterms:modified
030   dcterms:created
```

The internal field codes are referenced in `tab_fix_dc2usm` (or other `tab_fix` tables) which is used for conversion from DC to MARC 21-format (conversion step 2) or other internal Aleph metadata formats.

3.2.4 $data_tab/tab_fix (conversion of metadata)

The following entries should be defined in `$data_tab/tab_fix` if conversion of records from/to MARC/DC XML format or to other metadata formats (MAB, UNIMARC) are needed:

```
! 1                   2                                    3
!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
USMDC fix_doc_convit                  FILE=tab_fix_usm2dc
DCUSM fix_doc_convit                  FILE=tab_fix_dc2usm
USMUS fix_doc_convit                  FILE=tab_fix_usm2marcxml
USUSM fix_doc_convit                  FILE=tab_fix_marcxml2usm
```

Structure of the table:

Col. 1: Fix routine name

USMDC = Name of the fix routine to convert MARC-XML to Aleph-DC
DCUSM = Name of the fix routine to convert DC-XML to Aleph-MARC
USMUS = Name of the fix routine to convert Aleph-MARC to MARC-XML
USUSM = Name of the fix routine to convert MARC-XML to Aleph-MARC

Col. 2: Fix program name

`fix_doc_convit` is the program used.

Col. 3: Program arguments

FILE= Configuration table name
Name of `tab_fix` table which should exist in `$data_tab` of the library.

NOTE: In addition to `fix_doc_convit`, all programs which are defined for fix routine `INS` are used while loading new Aleph records from DigiTool.

If using Scenario #2 (e.g. Field INS with content INSERT) `$data_tab/tab_fix` in the BIB library should contain the fix program `fix_doc_001` in the INS section. This is to ensure a 001 field is created when the record is harvested.

3.2.5 $data_tab/tab_fix_dc2usm (conversion of metadata)

The table `tab_fix_dc2mab` contains rules for the conversion of internal field code into Aleph-MAB categories (conversion step 2). This description is valid for any configuration table which may be used together with `fix_doc_convit`.

Structure of the table:

Col. 1: Source Tag and Indicator

Col. 2: Source Subfield

Col. 3: Target Tag and Indicator

Col. 4: Target Subfield

Col. 5: Program Name with Arguments

```
! 1    2    3    4                5
```

```
!!!!!-!-!!!!!-!-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
#####   LDR     const_field       "00000^a^^^22^^^^^^^^4500",ONCE-IDN=01
#####   FMT     const_field       "CF",ONCE-IDN=02

017## a 001
001## a 245   a
016## a INS   a
```

⚠ Make sure that entries for Aleph categories **INS** and **001** exist in the table. Field **INS** is used as a flag to identify records that do not have a corresponding Aleph record in order to create a new bibliographic record.

NOTE: By default, `usm2us` and `us2usm` will map all tags "as they are" to and from MARCXML <-> Aleph-MARC with no changes—that is, 245 a to 245 a. Use the tab_fix tables only if non-default mapping is needed.

3.2.6 Parameters for service adam-08

The Aleph OAI harvester (`ue_13`) transfers data which have been provided from DigiTool OAI provider to the Aleph directory `$alephe_scratch`.
The file name is `ue_13_<date>.<seq.number>_<setname>.`

`ue_13` automatically activates Aleph service `adam-08`. This service is used to convert and load the records which have been harvested from DigiTool. `adam-08` is limited to loading/syncing objects which have Usage Type VIEW and their corresponding metadata provided in MARC 21 or Dublin Core XML format.

`adam-08` identifies the corresponding Aleph bibliographic records using a unique identifier (for instance, controlfield/001). Z403 records associated with Aleph bibliographic records are identified and updated if the DigiTool PID matches the entry in field `z403_note_5` (version 18+: z403_pid).

If no matching `001` is found, Aleph looks for an `INS` field with `content = INSERT`. This is Aleph's signal to create a new bibliographic record. Alternatively, a deleted record in DigiTool needs to have the delivery link from Z403 deleted in Aleph. Records that are indicated as deleted in the data `adam-08` receives from DigiTool are deleted according to PID – which is stored in Z403 for already synchronized records. If no matches are made for update/addition/deletion, the record is skipped and `adam-08` moves onto the next one.

Use the following command and parameters to run adam_08. Definitions for parameters follow (in the table below).

```
csh -f $aleph_proc/p_adam_08 <BIB library>,<input file>,<oai set
name>,<report file name>,<create thumbnail>,<create index>,<unique
identifier>,<fix procedure>,<cataloger name>,<cataloger level>
```

Parameter	Definition
Input File	Name of the input file which contains the digital entities provided from DigiTool. The input file must be under `$alephe_scratch`.
OAI Set Name	Name of the DigiTool OAI Set which includes the digital entities

Parameter	Definition
	to be loaded.
Report File Name	Name of the report file which will be created during the process, with information about the BIB records that have been updated. Default: <p_input_file>.doc_log
Create Thumbnail	Select if thumbnails should be created during the process. Relevant only if ADAM is fully licensed.
Create Index	Select if Full Text index files should be created during the process. Relevant only if ADAM is fully licensed.
Unique Identifier	Category of the bibliographic record which contains the unique identifier (must be 001).
Fix procedure	Definition of the fix procedure that is used for the conversion of the input data into Aleph-MARC (or MAB, UNIMARC (see above). Syntax: <format>:<fix_code>%<format>:<fix_code> format = format of the input data (can be "dc" or "marc") fix_code = name of the fix routine from $data_tab/tab_fix, col.1 % = "or" Example: `dc:DCUSM%marc:USUSM`
Cataloger Name	If the username of the cataloger is entered in this field, a `CAT` field containing the cataloger name is added to all records updated by the batch (either old or newly created). If the cataloger name is left blank, the `CAT` field will be created anyway, but it will contain only subfields $c (date), $l (active library), and $h (hour). If the cataloger name is set to "NO-CAT", `CAT` fields will NOT be added to the records updated or added.
Cataloger Level	This field can be used to determine the cataloger level that is recorded in the `CAT` field added to a record which has been updated or added.

3.2.7 Timestamps

Timestamps are counters in Oracle table Z52 (UTIL G/2):

```
ue13-last-date          - end date of last harvesting, YYYMMDD
ue13-last-time          - end time of last harvesting, HHMMSS
ue13-run-date           - date of last start of ue_13
ue13-run-no             - number of ue_13 starts
```

Last date and time will be used as parameters for the next harvesting. If re-harvest needs to start in the past, ue13-last-date and ue13-last-time can be set to the appropriate values. `ue_13` must be re-started to activate any new parameters.

3.2.8 Starting/Stopping ue_13

Once the above is setup, `ue_13` is ready to begin running. `ue_13` is a background daemon in each library and is run by using the libraries UTIL E menu.

To **start** `ue_13`—UTIL e 13

To **stop** `ue_13`—UTIL e 14

3.2.9 Tracking ue_13 and p_adam_08

The `ue_13` harvester log file is located in the `$data_scratch` directory of the Aleph library. It tracks both the harvesting of `ue_13` as well as the execution and result of `p-adam-08`. The log file follows the syntax:

```
run_e_13.<timestamp>
```

The harvested data—brought in from DigiTool by the `ue_13` service—is located in the `$alephe_scratch` directory and follows the syntax:
`ue_13_<datestamp>.<seq.number>_<setname>_001.tmp`

4 Aleph to DigiTool

The synchronization direction by which Aleph provides data to DigiTool requires setup on the DigiTool side (in terms of how to request and accept data) and on the Aleph side (in terms of how and what to provide to DigiTool).

4.1 Aleph

4.1.1 Aleph OAI providing data

Publishing-based Aleph OAI Data Provider exposes data prepared by the Aleph Publishing process. Refer to the Aleph Publishing documentation for more information about Publishing.

Aleph provides updated metadata to DigiTool by way of the OAI PMH protocol.

The Aleph OAI provider is configured under `./alephe/tab/oai/oaipubconf.xml`

The sets defined can be physical libraries (that is, `USM01`) or can be logical bases as defined in `tab_base` (for instance `USM01-ART`). Assuming the physical library in Aleph contains more records than are meant for synchronization with DigiTool, we recommend limiting the OAI providing set meant for DigiTool to a logical set.

Here is a sample OAI setup:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE oaiconf SYSTEM "oaiconf.dtd">
<oairoot>
<set>
  <setSpec>USM01-ART</setSpec>
  <setName>Art Images in DT</setName>
  <internalSet>USM01ARTD:oai_dc</internalSet>
  <internalSet>USM01ARTM:marc21</internalSet>
</set>
<repositoryName>Our Repository Name</repositoryName>
<baseURL>http://domain.server.com:8881/OAI</baseURL>
<adminEmail>ouremail@ourinstitution.com</adminEmail>
```

```
<description>
     <oai-identifier
         xmlns="http://www.openarchives.org/OAI/2.0/oai-identifier"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai-
identifier
            http://www.openarchives.org/OAI/2.0/oai-identifier.xsd">
        <scheme>oai</scheme>

<repositoryIdentifier>domain.server.com</repositoryIdentifier>
        <delimiter>:</delimiter>
        <sampleIdentifier>oai:domain.server.com:USM01-
000000001</sampleIdentifier>
      </oai-identifier>
</description>
</oairoot>
```

Follow any change to `oaipubconf.xml` with a re-start of the www server to activate.

You can access the OAI ListRecords for this set at your institution's Aleph OAI providing URL. For instance:
http://domain.server.com:8881/OAI?verb=ListRecords&metadataPrefix=marc21&set=USM01-ART

The OAI provider in Aleph allows requests to be made by date and time ranges based on the last updated date of the metadata records that make up any given set.

For instance, if record number 000001234 in Aleph has its MARC record updated on October 19, 2006 at 13:40, the following date range request should return that and any other matching updated records for the period.
http://domain.server.com:8881/OAI?verb=ListRecords&metadataPrefix=marc21&from=2006-10-19T00:00:00Z&until=2006-10-20T00:00:00Z&set=USM01-ART

DigiTool sends requests on a periodic basis to Aleph in order to sync up with the master catalog.

The format Aleph provides OAI data in may be in MARC 21 or in Dublin Core – depending on the preference for metadata storage on the DigiTool side.

4.2 DigiTool

4.2.1 Metadata (repository) synchronization

The repository synchronization provides one way for DigiTool to harvest metadata from an external OAI provider like Aleph for metadata enrichment/update of DigiTool objects. The relevant configuration file is located under

`./home/profile/synchronisations/conf/repository_synchronisation.xml`

Since the preferred matching identifier field within the metadata has to be indexed in DigiTool in order for the sync service to function, it may be necessary to adjust the

`repository_indexing_schema.xml` file with the appropriate metadata indexing definitions—not, however, with the digital entity indexing definitions stored in the same file.

NOTE: It is possible for the customer to add customer-specific synchronizations, including customer specific services as long as they are compliant with the basic outline of the synchronization and OAI standard.

4.2.1.1 Configuring the synchronization

The relevant setup for this functionality has to be performed under `./home/profile/synchronizations/conf`. The file `repository_synchronization.xml` is used to control the functionality. This file is divided into several sections, where each tag `<synchronization>` defines a separate OAI providing target to be queried and brought into DigiTool.

The following shows a sample configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<rs:synchronisations
xmlns:rs="http://www.exlibrisgroup.com/xsd/digitool/repository/synchronisation">
        <synchronisation name="dtlaleph" enable="true">
            <sources>
                    <source name="aleph"
class="com.exlibris.digitool.infrastructure.sync.SyncOAIAdapter">
                        <params>
                            <param
name="parser_class_name">com.exlibris.digitool.infrastructure.oai.OAIMARCSaxPars
er</param>
                            <param name="repository_name">OAI</param>
                            <param
name="base_url">http://domain.server.com:8881</param>
                            <param name="metadata_prefix">marc21</param>
                            <param name="from">2006-06-22T00:00:00Z</param>
<!--
                            <param name="until">2006-06-22T00:00:00Z</param>
-->
                            <param name="set">USM01</param>
                            <param name="scope">1440</param>
                            <param name="max_record_number">1000</param>
                        </params>
                    </source>
            </sources>
            <services>
  <service name="MaintenanceTool"
class="com.exlibris.digitool.repository.jobs.SyncMaintenanceTool">
                        <params>
                                <param name="job_name">Metadata Update</param>
                                <param name="unit">DTL01</param>
                                <param name="procToRun">update</param>
                                <param name="recordType">marc</param>
                                <param
name="recordIdentifier">/controlfield/001</param>
                                <param
name="mergeRules">md_merge_rules.xml</param>
                                <param name="isSA">true</param>
```

```
                        </params>
                        </service>
            </services>
            <scheduling sync_on_startup="true" type="every_x_hours" x="24"/>
        </synchronisation>
</rs:synchronisations>
```

The configuration of the above parameters will now be described in more depth.

4.2.1.2 Defining the external target source for synchronization

Every instance of `<synchronization/>` has two attributes: the name of this synchronization and the enabled attribute—either `true` or `false`.

To activate a synchronization, the enabled attribute must be set to `true`.

The `source` section holds the target information for the OAI-Request to the external OAI provider (Aleph) system. This `source` element has two attributes: the name of the source and the connected java-class that runs the harvest of the metadata. The parameter section comes after the source element. The `<params>` element holds several sub-elements named `<param>`. Each `<param>` element has an attribute "name" which stores a defined value.

NOTE: The configuration file is read upon startup of the DigiTool server. Any changes to this file will only take effect after restart.

The following parameters can be configured:

- parser_class_name: a java-class to parse the metadata.
  Sample:
  ```
  <param name="parser_class_name">
  com.exlibris.digitool.infrastructure.oai.OAIMARCSaxParser
  </param>
  ```

- repository_name: name of the target repository OAI program
  Sample – for Aleph:
  ```
  <param name="repository_name">OAI</param>
  ```

- base_url: the URL and port of the source from which the data should be harvested – in many cases, Aleph.
  Sample:
  ```
  <param name="base_url">http://domain.server.com:8881</param>
  ```

- metadata_prefix: the metadata type in which the incoming metadata is described – OAI-PMH standard (marc21/oai_dc).
  Sample:
  ```
  <param name="metadata_prefix">marc21</param>
  ```

- from: start date and start time for looking up OAI provided metadata updates – in UTC. The definition here is only active, if no valid timestamp file under

`j_home/profile/synchgronisationa/timestamps` exists – usually the first run only.
Sample:
```
<param name="from">2007-06-22T00:00:00Z</param>
```

- until: end date and end time. This optional parameter allows a synchronization to be run within a defined range of time - UTC. Otherwise this parameter should be commented out and each run will be based on the "Scope" definition.
  Sample:
  ```
  <param name="until">2007-06-22T00:00:00Z</param>
  ```

- set: name of the source set. According to the OAI-PMH standard.
  For Aleph, the value here is according to the value given in the setspec-element of the `oaipubconf.xml` (the configuration file for OAI data provider setup on the Aleph side).
  Sample:
  ```
  <param name="set">USM01-ART</param>
  ```

- scope: defines the range of time to search the OAI provider with each successive sync run. It essentially becomes the "until" date and time for each run = timestamp or initial from date + scope time. The value is set in minutes.
  Sample (24 hours in minutes = 1440):
  ```
  <param name="scope">1440</param>
  ```

- max_record_number: defines the maximum number of records on which one single run should be limited. If there are more records than allowed
  Sample:
  ```
  <param name="max_record_number">1000</param>
  ```

4.2.1.3 How to set up the service – updating DigiTool metadata

For each synchronization, one or more services can be configured. Each service element has two attributes: `name` is the name of the service and `class` is the called java-class for the procedure. The services are used for defining the further processing of the harvested data. The following parameters are configurable:

- job_name: name of the job which should run on the metadata.
  Sample:
  ```
  <param name="job_name">Metadata Update</param>
  ```

- unit: the administrative unit in DigiTool to which the object and the relevant metadata to be updated belongs to:
  Sample:
  ```
  <param name="unit">DTL01</param>
  ```

  For any unit:
  ```
  <param name="unit">ALL_UNITS</param>
  ```

- identifier: the identifier field in the DigiTool stored metadata used to match against the harvested metadata. This may be any field, even a user-defined one.
  Samples:

```
<param name="identifier">/controlfield/001</param>

<param name="identifier">dc:alephsyncid</param>
```

- procToRun: The procedure to run for the data brought in – possible options:
  1. **update** – that is, only updates based on matching identifiers found both in the data brought in and the data in the DT repository.
  2. **new** – that is, updates will occur like in #1, but in addition, any record brought in from the external source which doesn't have a matching identifier will be added to the DigiTool system as a new MID (metadata identifier). Any added MIDs can be used in the DigiTool system and associated with objects PIDs already in the system or to be loaded in the system in the future.

- mdName: the descriptive name of the metadata stored in DigiTool for example, marc, dc
  Sample:
  `<param name="mdName">marc</param>`

- mergeRules: Optional setting for specific fields that should be saved or deleted from the original metadata record in DigiTool or in the metadata brought in from the external source. Default is to override the metadata in DigiTool with the entire master record being pulled in. The template(s) for md_merge_rules.xml are stored under
  `./home/profile/conf/templates/metadata/.`
  Sample:
  `<param name="mergeRules">md_merge_rules.xml</param>`

  Here is a sample `md_merge_rules.xml` definition:

```
<?xml version="1.0" encoding="UTF-8"?>

<mr:md_merge_rules
xmlns:mr="http://com/exlibris/digitool/common/formatting/xmlbeans">
  <fields>
    <field action="delete">
      <location type="md" md_name="descriptive" md_type="marc"
path="MID/#/#/a"/>
      <location type="md" md_name="descriptive" md_type="dc"
path="//dc:mid[not(@*)]"/>
    </field>
    <field action="insert" force="false">
      <location    type="md"    md_name="descriptive"    md_type="marc"
path="leader"/>
      <location type="md" md_name="descriptive" md_type="marc"
path="245/#/#/a"/>
      <location type="md" md_name="descriptive" md_type="dc"
path="//dc:title[not(@*)]"/>
    </field>
```

```
  </fields>
</mr:md_merge_rules>
```

The above `md_merge_rules.xml` definition will delete the fields MID in any incoming MARC record and dc:mid in any incoming dc record.

NOTE: Deletion of the leader (action=delete) is not supported for the leader field.

The `md_merge_rules.xml` configuration file is located under the `./home/profile/conf/templates/metadata/` directory.

Here are some examples of correct syntax necessary to address different kinds of fields within a `<field>` section in the `md_merge_rules.xml`:

Example: MARC leader field

```
      <location type="md" md_name="descriptive" md_type="marc"
path="leader"/>
```

Example: standard MARC field

```
      <location type="md" md_name="descriptive" md_type="marc"
path="245/#/#/a"/>
```

Example: MARC Controlfield

```
      <location    type="md"    md_name="descriptive"    md_type="marc"
path="/controlfield/008
 "/>
```

Example: Dublin Core field – type: element without Encoding Scheme

```
      <location type="md" md_name="descriptive" md_type="dc"
path="//dc:title[not(@*)]"/>
```

Example: Dublin Core field – type: element with Encoding Scheme

```
      <location type="md" md_name="descriptive" md_type="dc"
path="//dc:subject[@xsi:type='dcterms:LCSH']” / >
```

The available actions are:

delete - detach records from input record.

insert - maintain tags from the DigiTool record to add them to the input record.

The update is matched either on the record identifier or the MID (in the case of the first Aleph back-sync to DigiTool—that is, no record identifier yet exists, but rather only MID).

NOTE: When the tag already exists in the input record:

- With force option set to false, no insert will be performed.

- With force option set to true, the DigiTool tag will be added to the input record.

To enhance the performance of the metadata update job, the job is performed outside JOBSS by setting this parameter:

```
<param name="isSA">true</param>
```

The scheduling of the sync is defined in the following line:

```
<scheduling sync_on_startup="true" type="every_x_hours"
x="24"/>
```

Available types are:

- Hours' interval – put "every_x_hours" for the "type" attribute, the value of the "x" attribute will define how many hours between synchronizations.
  In the following example, synchronize every 48 hours:

  ```
  <scheduling type="every_x_hours" x="48"/>
  ```

- Minutes' interval – the value of "type" is "every_x_minutes" and the value of "x" is the number of minutes between synchronizations.
  In the following example, synchronize every 30 minutes:

  ```
  <scheduling type="every_x_minutes" x="30"/>
  ```

- Certain time daily – Configure the exact hour on which the replication will run every day. "x" for the hour, "y" for the minutes.

  In the following example, synchronize every day at 2:30 am.

  ```
  <scheduling type="every_day_at_x_and_y" x="2" y="30"/>
  ```

NOTES

The scheduling intervals are relative to the DigiTool server startup time.

Avoid using small intervals except in cases of very small synchronizations. If you can identify the most idle hours of the day of the server, use the specific-hour-scheduling for those times.

The service can run automatically using the configuration information or can be started manually via the "/mng" management module – Maintenance jobs – Metadata Update.

4.2.1.4 Running the Metadata update manually

Generally, this service runs as part of the synchronization itself and does **not** need to be run manually.

However, the update of DigiTool metadata can be performed using the maintenance job—Metadata Update—rather than or in addition to the automatic definitions in `repository_synchronisation`. The parameters are identical to those that can be defined in the `repository_synchronisation.xml` file, but instead, the parameters

here are defined in the standard `repository_jobs_configuration.xml` file – in `j_conf`.

The input file is the name or full path of a metadata xml file that has been harvested into the "files" directory under `./home/profile/synchronisations/`.

4.2.2 Tracking the synchronization

4.2.2.1 Logs

Repository synchronization logs can be found under `./home/profile/synchronisations/logs`.

The general log, called "`manager.log`", is created on startup and gives information on the synchronization scheduling, general information and general initialization errors.

The detailed synchronization log (created for each synchronization run) is a dedicated log following this naming convention:
`<synch-name>_<schedule-id>.log.<log-id>`.

`<synch-name>` is the value of the "name" attribute in the "synchronization" tag.
`<schedule-id>` is the id of the scheduling for the synchronization
`<log-id>` is an incremental number that indicates the index of the file for the same log.

The logs give information about phases the synchronization process is in, number of files added/updated in each phase, time it took for each phase to complete and any errors that may have occurred while running.

4.2.2.2 Timestamps

Every time the synchronization starts, it looks for a timestamp file in the `<synchronization_home>/timestamps` directory. This timestamp represent the last exact time the synchronization was run, and it is used to check for changes since that time. When it finishes synchronizing, it updates the timestamp file. If there is no timestamp file found, the date and time defined in the `repository_synchronisation.xml` in the parameter "from" under the source element is taken as the new initial timestamp.

You might want, for example, to run a new synchronization, but you are not interested in objects from last year. Entering a timestamp file before starting the server (with content such as `2006-01-01 00:00:00`) will allow this to occur.

Alternatively, if you would like to start over, removing all timestamp files will begin the synchronization from the initial setting in the configuration file.

The relevant timestamp can easily be found by its name, which is the synchronization name with the "timestamp" extension, for example, `synchTest.timestamp`.

4.2.2.3 Files

The files directory holds the metadata xml files which are harvested from the external OAI provider. For each synchronization run (which retrieves data) a file with all metadata sets is created.

The naming convention is as follows:
`<sync_source_name>_0_<yyyymmdd.hhmmss>_0.xml`

These files are taken as input for the process defined by the service-section in `repository_synchronisation.xml`. The input will be the name of the xml file holding the harvested metadata – and will be checked against existing repository indexed data in order to update matching records with this harvested metadata.

4.2.2.4 Run

The run folder holds the basic parameters which were defined in the `repository_synchronisation.xml` file – denoting how to update DigiTool objects with the contents of the harvested metadata files. It is created automatically and does not need to be configured.

4.2.2.5 Configuring the repository indexing schema

For the functionality of the synchronization, the identifier field chosen for matching between systems has to be indexed in the `repository_indexing_schema.xml` under the `./system/conf` directory. This has to take place under the relevant metadata indexing section for Dublin Core or MARC—not the Digital Entity indexing section.

For example, using MARC controlfield 001 defined under the Marc metadata index section:

```
<field index_enabled="true" search_enabled="true" ui_code="210"
ui_default_text="001"
index_code="4010" normalizing_profiles_ref="generic">
      <field_path>/controlfield/001</field_path>
 </field>
```

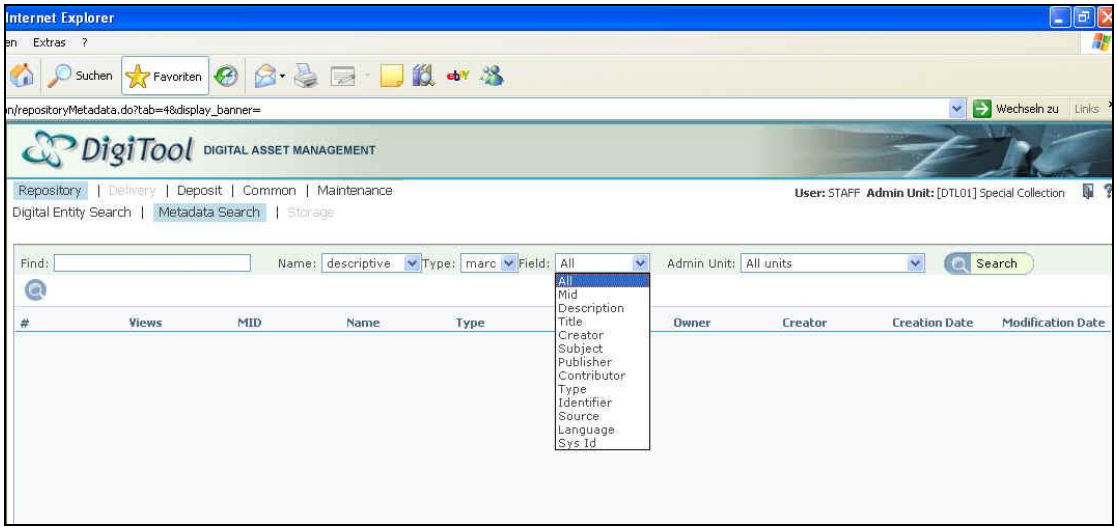
Using a local field, 988 subfield a (MARC):

```
<field index_enabled="true" search_enabled="true" ui_code="210"
ui_default_text="SysId"  index_code="4010"
normalizing_profiles_ref="generic">
      <field_path>988/#/#/a</field_path>
</field>
```

The value for the `index_enabled` and `search_enabled` attribute has to be `true`. The `ui_code` and `index_code` is consecutively numbered. The `ui_default_text` is the displayed text in the repository metadata search (should not contain spaces). The field which should be indexed has to be defined in the `<field_path>` tag.

NOTE: Re-loading repository configuration is necessary for activating any changes. Additionally, for already existing repository data, re-indexing metadata may be necessary if the field being used for identifying/matching is not indexed.

The new indexed field should be displayed in the management module in the repository **metadata search** when the relevant descriptive metadata is chosen. If you search by your preferred "sync" field under the Metadata search and do not find results, neither will the metadata synchronization services. Ensure the index is defined properly, re-loaded (or re-start to JBOSS) and data re-indexed.



5 Aleph Enrichment

An additional possibility for updating records in DigiTool with Aleph metadata updates, but not on a synchronized schedule, is using the maintenance job "Aleph enrichment." Aleph enrichment works differently from the synchronization service in that it currently supports only MARC metadata—and expects that the matching identifier will be the controlfield 001. Additionally, updates by date range are not requested from Aleph, but rather, DigiTool gets a set of PIDs to update in DigiTool, reads the metadata and finds any 001 fields present in the metadata and sends individual OAI requests—GetRecord and NOT ListRecords—for those particular 001 fields. The 001 fields are expected to be equal to the record number in Aleph.

The Aleph enrichment service is less sophisticated than the synchronization service but is also easier to track and set up. If there are not many updates expected in Aleph, the enrichment service may be a preferable option over the synchronization "Metadata Update" service for bringing in the Aleph master catalogued records into DigiTool.

5.1.1.1 Running Aleph enrichment

You can run Aleph enrichment from the maintenance jobs in the "mng" module.
The parameters require you to choose a set of PIDs by filling in the job form. Specific parameters for this service are:

Data Source: Data source defines which Aleph server to request records from by way of the X-Server `find-doc` function. The possible data sources are defined in `global.properties`:

```
aleph.server=domain.server.com
aleph.port=8882
aleph.base=usm01
```

They are used in `repository_configuration.xml.tmpl`:

```
   <data_sources>
                <data_source transport_protocol="x-server">
                        <id>1</id>
                        <name>Aleph Source</name>
                        <adapter_class/>
                        <data_source_parameters>
                                <data_source_parameter name="url">

http://@@aleph.server@@:@@aleph.port@@/X?op=find-do
c&amp;base=@@aleph.base@@
                                </data_source_parameter>
                        </data_source_parameters>
                </data_source>
        </data_sources>
```

After running `set_globals.sh` in `repository_configuration.xml`, they appear as follows:

```
   <data_sources>
                <data_source transport_protocol="x-server">
                        <id>1</id>
                        <name>Aleph Source</name>
                        <adapter_class/>
                        <data_source_parameters>
                                <data_source_parameter name="url">

http://domain.server.com:8881/X?op=find-do
c&amp;base=usm01
                                </data_source_parameter>
                        </data_source_parameters>
                </data_source>
        </data_sources>
```

Save Tags: Similar to `md_merge_rules` in the metadata update/sychnronization service, but only involves keeping metadata fields in the MARC record stored in DigiTool prior to running the enrichment service. For instance, choosing `245/#/#/a` in `Save Tags` will keep the DigiTool `245a` tag, but the rest of the metadata record from Aleph—if 001 matches—will be brought in to override the DigiTool record.