Using open source tools to expose cross collection data in the LIDO schema

Part II: David Parsell

Overview

Adoption of the LIDO schema worldwide has opened up many data sharing opportunities for the Yale Center for British Art, which has long sought to expose descriptive collection and research data to museums, researchers and data aggregators.

Optimally, the YCBA prefers to use open source tools when building information systems, and to that purpose turned to the OCLC Mellon-funded Museum Data Exchange project that had built an open source data export tool using the proprietary software; COBOAT by Cogapp. The OCLC project focused on exporting data from TMS, the Gallery Systems collection management system, but the software is adaptable to other collection management systems.

The OCLC Museum Data Exchange project can be found at http://www.oclc.org/research/activities/coboat/default.htm .

COBOAT is a metadata publishing tool developed by Cognitive Applications Inc. (Cogapp) that transfers information between databases (such as collections management systems) and different formats, but it is not an open source tool. OCLC commissioned Cogapp to extend COBOAT to enable the extraction of collection data in the CDWA Lite schema and to make COBOAT available under a fee-free license for the purposes of publishing a CDWA Lite repository of collections information from museums using TMS.

The configuration files available with the COBOAT download from OCLC are designed for use with the Gallery Systems TMS collections management system. However, you can customize the configuration files to adapt COBOAT to different vendor-based or homegrown database systems of collections information.

The OCLC COBOAT project was completed in 2008 and designed specifically for TMS collection management systems. In 2010, the YCBA implemented COBOAT/ OAIcatmuseum to expose collection data in the CDWA Lite schema, but when the much richer LIDO schema superseded CDWA Lite, the YCBA choose to move to the LIDO schema.

Given the successful YCBA CDWA Lite experience with COBOAT and OAIcatmuseum, the obvious upgrade path was to explore extending COBOAT and OAIcatmuseum to support the LIDO schema. Fortunately, the remarkable flexibility of COBOAT allowed a fairly painless extension to the LIDO schema.

In the fall of 2011, the YCBA, Cogapp and OCLC, extended COBOAT and OAIcatmuseum to facilitate exposing LIDO schema XML records for harvesting via the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)

COBOAT and OAIcatmuseum are “open source” software available on the OCLC website to any museum requesting the software. However, COBOAT requires a free license from Ben Rubenstein at Cogapp benr@cogapp.com. Cogapp has extended the original CDWA Lite license to include the LIDO schema.
http://www.oclc.org/research/activities/coboat/default.htm.

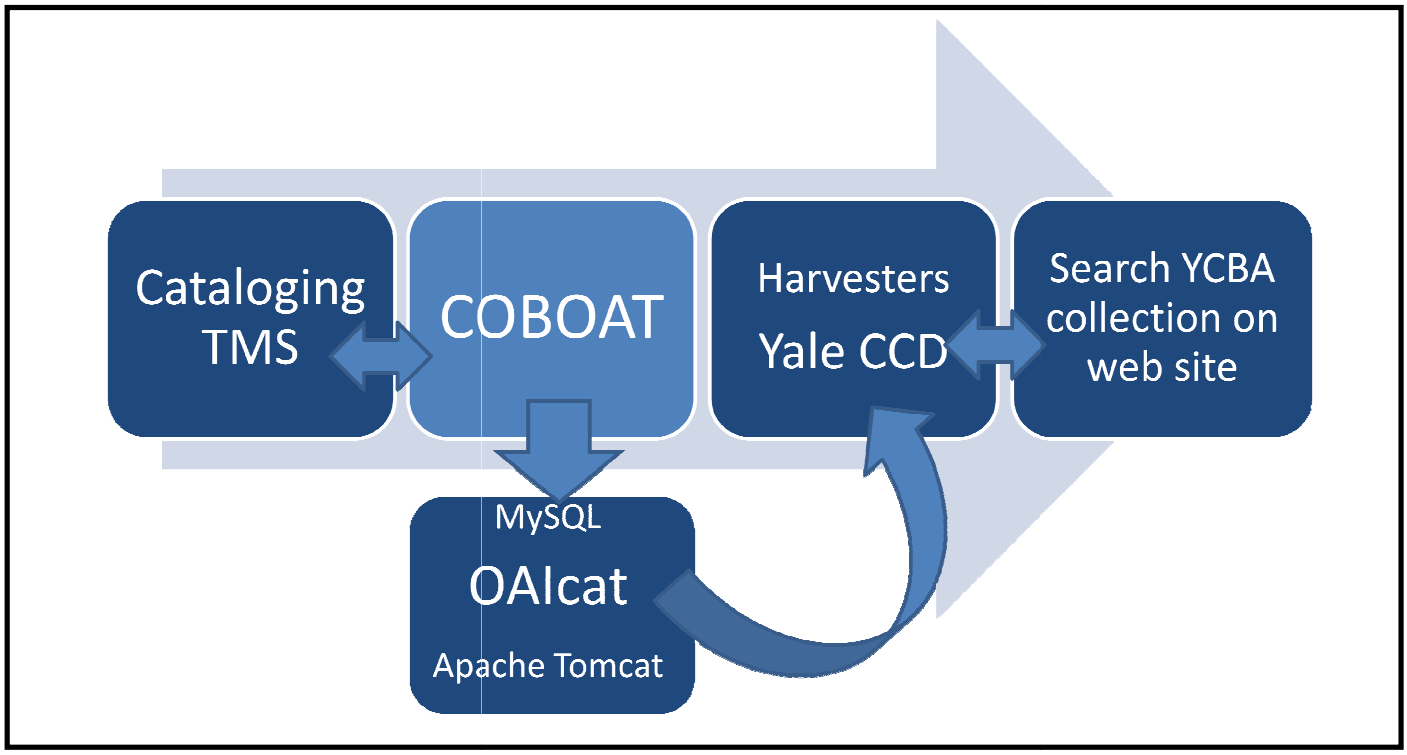
Besides COBOAT and OAIcatmuseum, you will need the following open source software products to support the data harvesting.

1. MySQL database software. COBOAT sends the xml documents to the database for harvesting.
2. MySQL workbench; a GUI interface which expedites reviewing the xml documents in MySQL.
3. Apache Tomcat to enable harvesting through OAIcatmuseum.
4. A good text editor for writing the COBOAT scripts. I used the open source Notepad++, but Oxygen also works well.
5. Software to validate the xml schemas. The YCBA uses Oxygen, which must be purchased.

The COBOAT and OAIcatmuseum software downloads include comprehensive manuals providing excellent guidance for the installation, coding and operation of the software.

This document goes beyond the manuals; exploring the actual YCBA implementation and challenges to extending COBOAT and OAIcatmuseum to expose data in the LIDO schema.

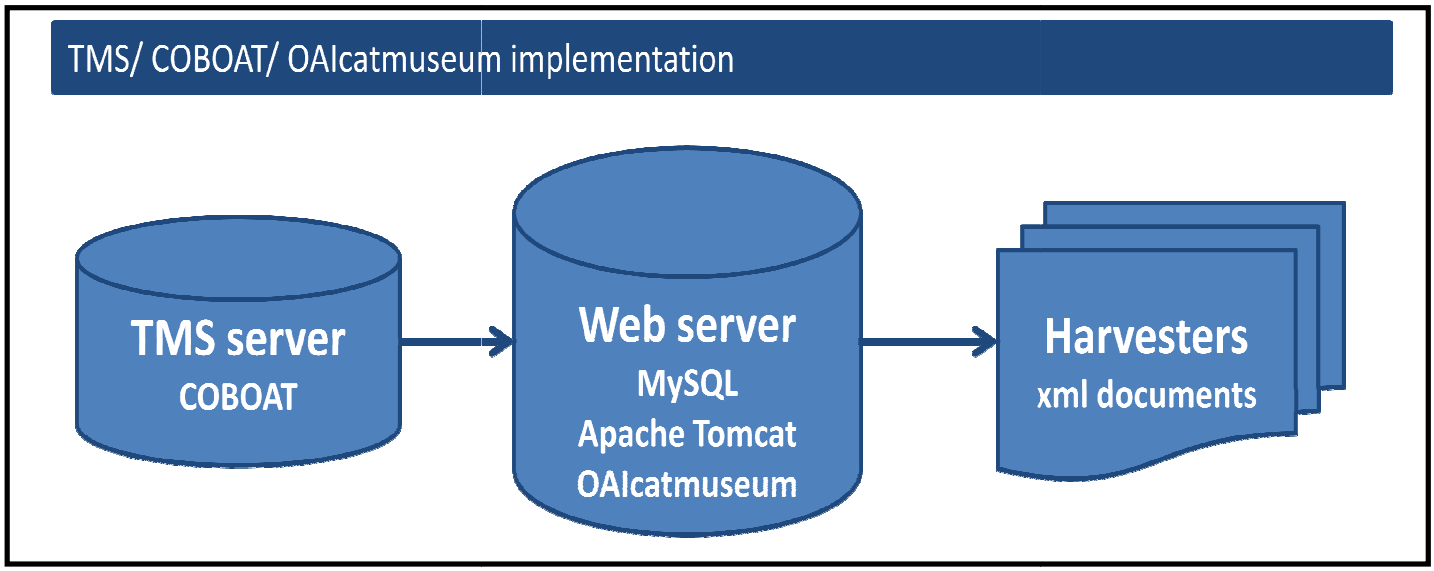
Using open source tools to expose cross collection data in the LIDO schema



Functionality

COBOAT and OAIcatmuseum together perform only one task; extracting data from a collection management system and generating xml documents that are exposed for harvesting.

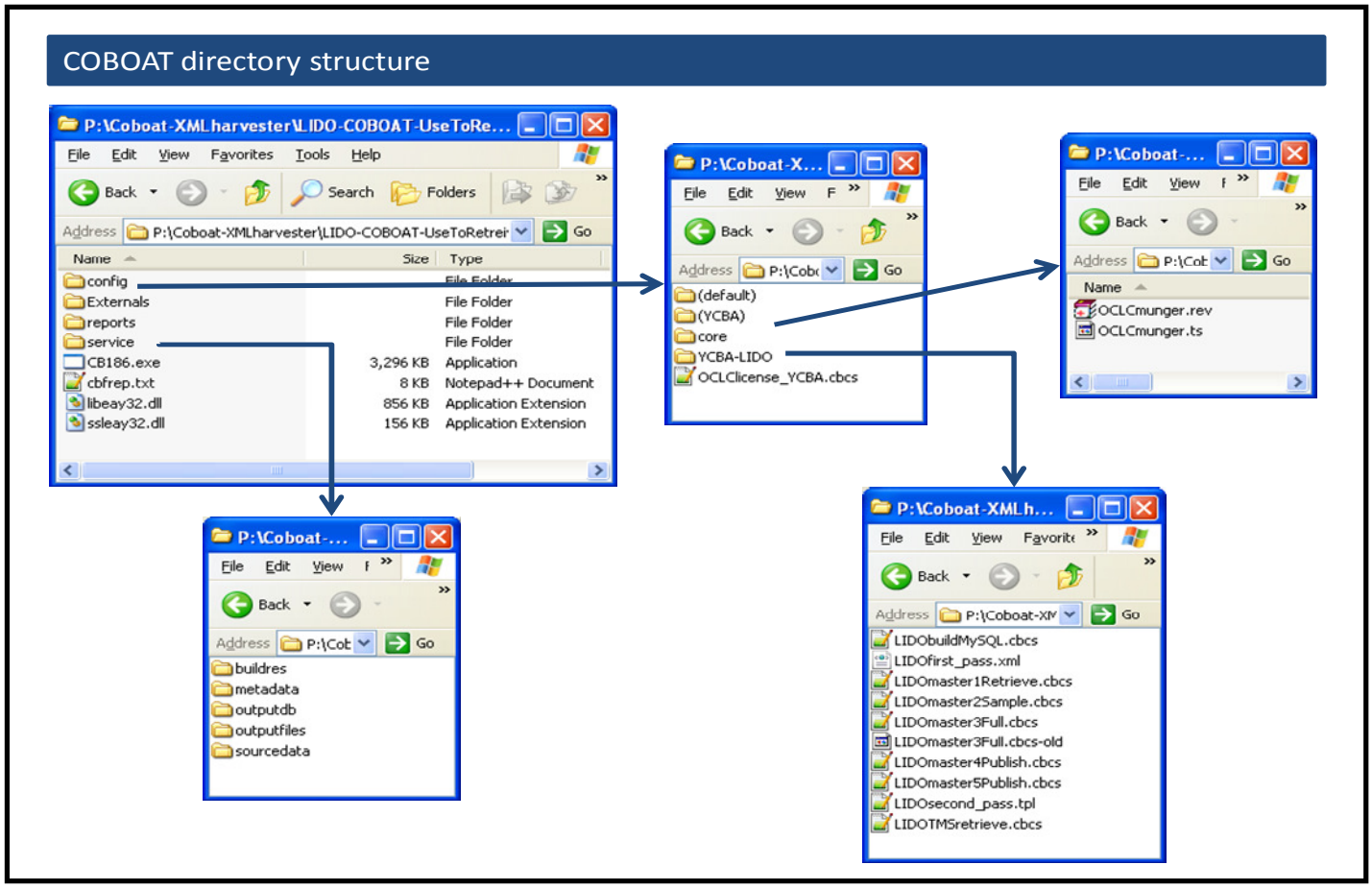
COBOAT does 80% of the work. It exports the data from the database, generates xml records from the data and delivers the xml records to another database where OAIcatmuseum exposes the xml records to harvesters.



YCBA Implementation

Using open source tools to expose cross collection data in the LIDO schema

COBOAT and OAIcatmuseum can be set up in a number of different configurations. YBCA used a two server configuration based on security concerns. To avoid intrusions in the production TMS database, COBOAT was installed on the TMS production server while OAIcatmuseum and the MySQL database were installed on a 2nd server dedicated to harvesting.



COBOAT directory structure

The COBOAT executable (CB186.exe), dlls and a copy of the log file reside in the root or top folder.

The three customizable scripts needed to run COBOAT are located in a museum specific subfolder under the CONFIG folder, I. E. YCBA-LIDO. COBOAT ignores folder names in () when building the COBOAT menu. Your COBOAT license must be placed in the active museum specific script folder.

The three customizable scripts used by COBOAT perform specialized tasks.

1. The TMSretrieve script exports data from the collection and stores it in text files in preparation for building the xml documents. I chose to write the export queries in Transact-SQL. All of the pre-processing logic is in this script. I used Notepad++ to write and edit the scripts.

2. The 1st_pass script takes the data from the text files and sets it up in memory arrays in preparation for building the xml documents. This script can also be used to modify/ filter the extracted data, but for simplicity I choose to put all of the data pre-processing logic in the TMSretrieve script.

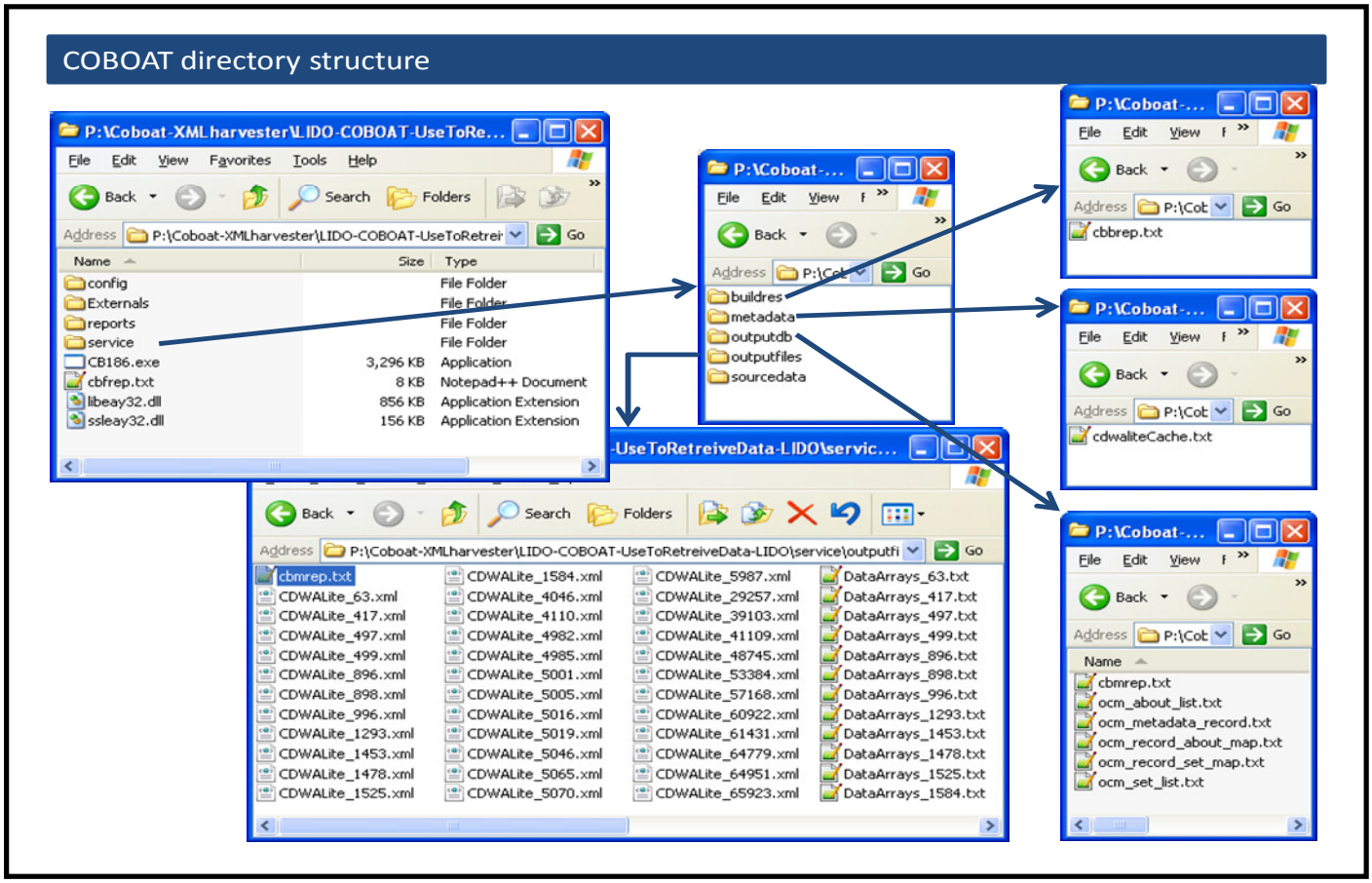
3. The 2nd_pass script is the xml document template used by COBOAT to generate the xml documents. COBOAT does not validate schemas, so any schema and elements can be used. I used Oxygen to validate the template.

The LIDObuildMySQL.cbcs script is also in the active museum specific folder. It is the configuration file for COBOAT processing. I made a few changes to this file to adjust the munge processing for LIDO as will be discussed later.

CORE folder contains the compiled COBOAT munger.

EXTERNALS folder contains COBOAT dlls.

REPORTS folder contains error reports listing missing data in the generated xml documents. The reports are useful to review the missing data by data type. ObjectIDs are listed in the reports, allowing you to identify objects requiring further cataloging.



SERVICE folder has a number of sub-folders for storing the data files generated during COBOAT processing.

BUILDRES folder maintains a copy of the latest log file also found in the root COBOAT folder.

METADATA folder contains a file with only two fields; checksum and last build date. They are used to track xml document changes. COBOAT only sends a new xml document to OAIcatmuseum if the checksum changes, indicating that the object record data changed.

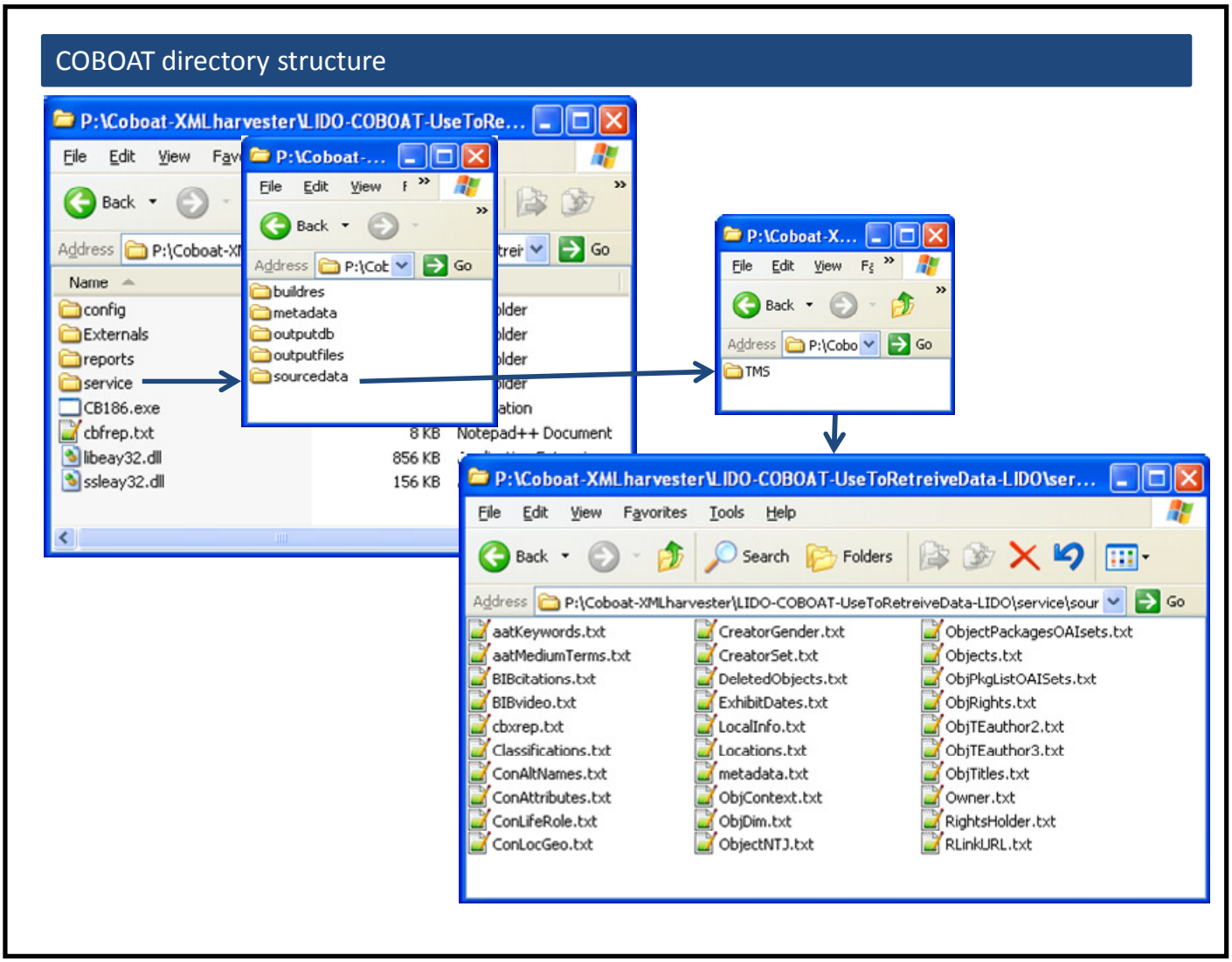
OUTPUTDB folder a contains a copy of the xml documents sent to the main MySQL table, as well as the supporting data stored in the other MySQL OAIcatmuseum tables.

OUTPUTFILES folder contains sample xml documents and data arrays that can be generated when testing the xml template. These are very useful for finding errors before generating all of the xml documents. You can select the objects that you want to review during testing by entering the appropriate objectIDs in the “Generate file samples” menu text file.

SOURCEDATA folder→TMS folder contains the text files exported from TMS by the “TMSretrieve” script. This folder also contains text files from other sources to be used in the xml munge. I.E. the YCBA includes a data file with pointers to the object records in the YCBA searchable on-line collection.

NOTE: the YCBA does not use COBOAT to extract image files from TMS although it is possible. Please refer to the COBOAT documentation.

Using open source tools to expose cross collection data in the LIDO schema



COBOAT was written to create xml documents in the CDWA-Lite schema, but the software has proven to be easily adapted to creating xml documents in the LIDO schema.

However, LIDO did introduce complications that required OAIcatmuseum modification so that a LIDO header would be attached to each LIDO xml document instead of the default CDWA Lite header. OCLC made the modification to OAIcatmuseum and has released v1.1 on the OCLC website. The new version will allow you to expose data with either the CDWA-Lite or LIDO headers. You control the choice in the OAIcatmuseum configuration file discussed later.

COBOAT will run on Macintosh or Windows operating systems. COBOAT has a small footprint which includes the executable and a few support files in subfolders. Most of the COBOAT space requirements are devoted to the COBOAT generated data files and xml documents, which may rule out a workstation in favor of using a server with sufficient disk space. The total YCBA production COBOAT software and data storage is approx. 480mb for 8,093 records.

COBOAT does have a few restrictions; COBOAT has a compiled program that reads the exported data and generates the data in xml documents (munge function). The compiled program cannot be changed without contracting with Cogapp.

If the xml documents are going to be exposed and harvested using OAIcatmuseum, each xml document will only display either a CDWA-Lite or LIDO header depending on the OAIcatmuseum configuration.

Using open source tools to expose cross collection data in the LIDO schema

Running COBOAT manually on a Windows server requires access to the server via “remote desktop”. COBOAT can be set up to run automatically on the server using Windows “task manager”. COBOAT can be run manually from a workstation via a mapped drive to the server, but this method dramatically increases the processing time (7min to 33min for just TMSretrieve).

COBOAT Windows/ SQL server setup

The YCBA uses Windows server (2008r2), with SQL server (2008r2), supporting the TMS database, and Windows XP on the TMS workstations. The COBOAT setup parameters will change depending on the operating system and collection management database used by other institutions.

COBOAT uses an ODBC connector to the TMS database. “tms4coboat” is the recommended name, but any name can be used as long as the ODBC name in the COBOAT script matches. Note; you need to use the 32 bit ODBC manager on the Windows server to set up the ODBC connection, not the 64 bit ODBC manager. C:\windows\syswow64\odbcad32.exe.

A SQL server login with SQL server authentication is required to get past SQL Server forcing a password change every time COBOAT tries to connect to TMS with a Windows authenticated login. Use SQL server to create the SQL login and password. SQL server will only allow a SQL login to programmatically probe SQL server.

This following SQL script is run daily by the SQL agent to reset the COBOAT SQL login password.

exec sp_change_users_login 'auto_fix', 'SQLlogin', NULL, 'password'

A MySQL login and password is required to allow COBOAT to connect to MySQL to refresh the OAIcatmuseum database with the latest xml documents. The login is set up in MySQL using Workbench and rights are granted to the OAIcatmuseum database that will store the xml documents.

Incremental export to OAIcatmuseum

By default, COBOAT performs an incremental export of the xml documents to OAIcatmuseum. COBOAT always extracts and processes the complete TMS data set as defined in the TMSretrieve script, but overwriting the existing xml document and delivery to OAIcatmuseum is controlled by the checksum history file in the METADATA folder.

When COBOAT builds the xml documents, it compares the new document checksums to the stored checksums from the previous export. If the checksums match for the same objectID, COBOAT sends the stored xml document with the previous build date to OAIcatmuseum. If the checksums are different, COBOAT sends the new xml document with the new build date to OAIcatmuseum and overwrites the older xml document, checksum and date in the METADATA and OUTPUTDB directories.

Harvesters can do an incremental retrieval by adjusting their retrieval software to harvest within a date range, or they can ignore the date range function and perform a full harvest.

The workaround to guarantee sending all of the xml documents to the OAIcatmuseum is to delete the checksum history file in the METADATA folder, forcing COBOAT to rebuild the file and to send all of the new xml documents to OAIcatmuseum.

Deleting released records from external databases

This functionality is designed to identify records that harvesters should remove from their external databases when the data provider has decided to remove the record from circulation.

COBOAT and OAIcatmuseum initially did not have the delete functionality. Cogapp and OCLC modified their respective software and implemented the delete functionality in April, 2011.

To delete a released record, the data provider enters the object record into a TMS “deleted records” object package. COBOAT reads the “deleted records” array during the munge function and sends the “deleted” objectID and a blank xml record to OAIcatmuseum. The blank record is harvested and eventually overwrites the object record in the harvesters’ database, effectively deleting the withdrawn object record. The blank deleted records need to be continuously exposed by COBOAT and OAIcatmuseum to ensure deleting the records from all external databases.

Using open source tools to expose cross collection data in the LIDO schema

COBOAT/ OAIcatmuseum changes needed to support the LIDO schema

While the CDWA-Lite schema worked very well with the default COBOAT settings, a number of COBOAT processing parameters needed to be changed to adapt COBOAT to the much richer LIDO schema and larger xml document.

The only CDWA-Lite change needed was modifications to COBOAT and OAIcatmuseum to activate the “xml document delete functionality”.

However, once COBOAT was extended to generate LIDO schema xml documents, a number of parameter changes were required to accommodate the much larger and more complex LIDO xml document.

The average CDWA Lite document size is 7 – 9 kb. The average LIDO document size is 26 – 38 kb.

The much larger LIDO xml documents forced changing the processing parameters from the default;

chunkrows="100" chunkinput="10000" changed to chunkrows="10" chunkinput="10".

On the production TMS server, no speed difference between 10 or 100, but the smaller dev server only works when set to 10. Conversely, a similar change to the TMSretrieve processing from 10000 to 10 significantly slowed down the processing.

The processing changes had the added benefit of processing the data much faster. Processing the original LIDO documents with the default COBOAT parameters took almost 120 minutes total. Changing the above parameters to accommodate larger LIDO xml documents dropped the processing time to approximately 12 minutes.

COBOAT production processing is done on a dual CPU VM with 4 GB memory and a gigabit network while the dev server is a single CPU VM with 2 GB memory. The parameter changes were made in the LidoBuildMySQL.cbcs file. A MySQL timeout error message prompted the processing parameters changes.

Eventually, as the YCBA continued to include additional data in the LIDO schema, COBOAT stopped working with an error message stating that the xml document was too big to be inserted in the MySQL db.

This problem was resolved by changing the field type from text to longtext to accommodate the very large LIDO xml documents; again the parameter is in the LidoBuildMySQL.cbcs file.

Text field size = 64k bytes, longtext field size = 4gb bytes.

I had to adjust COBOAT to process Latin-1 data during the munge process. The COBOAT documentation states that the UTF-8 conversion would be handled by OAIcatmuseum, which was fine initially, but as large fields of text were included in the xml documents, random blocks of text would cause errors.

I turned on the UTF-8 conversion prior to the munge process to handle the errors caused by the characters in large text fields. OAIcatmuseum still does the UTF-8 conversion, but it is too late in the process to handle the errors I encountered. This is the setting in the buildmysql.cbcs file.

```
convertcharset from="entities" to="utf8"
```

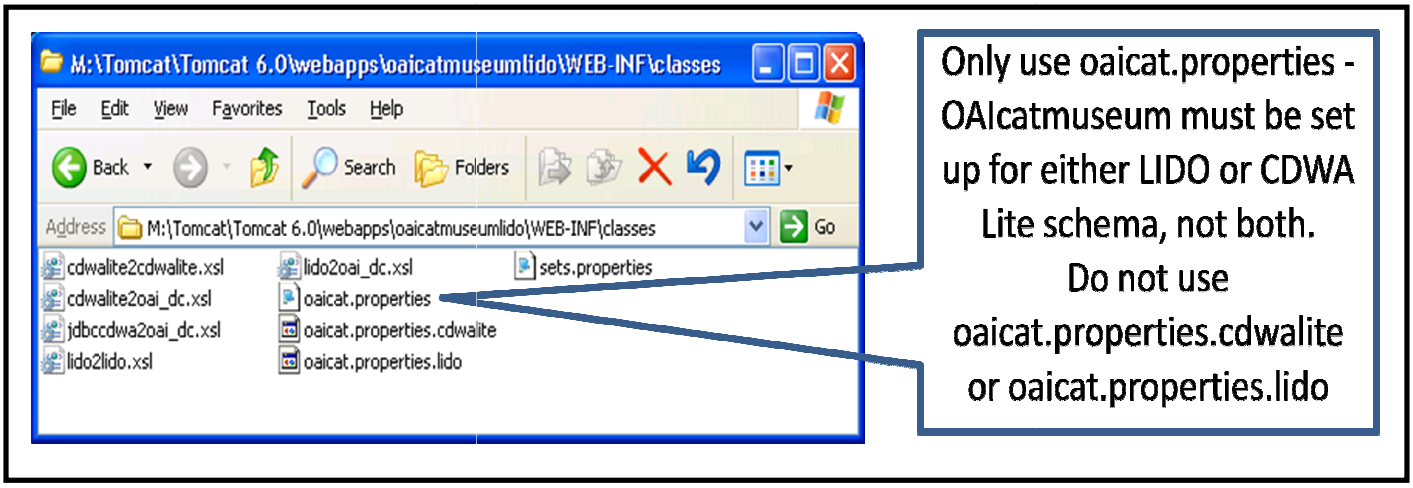
Finally, unlimited size text fields are a problem for COBOAT, which could not process them during the TMSretrieve function. The problem was solved by using a field type conversion in the TMSretrieve script. Following is an example of the conversion code. The large field size specified in the conversion is not a problem as long as the field is defined with a set size.

```
convert (varchar(8000),textentries.TextEntry) as Lettering
```

OAIcatmuseum implementation

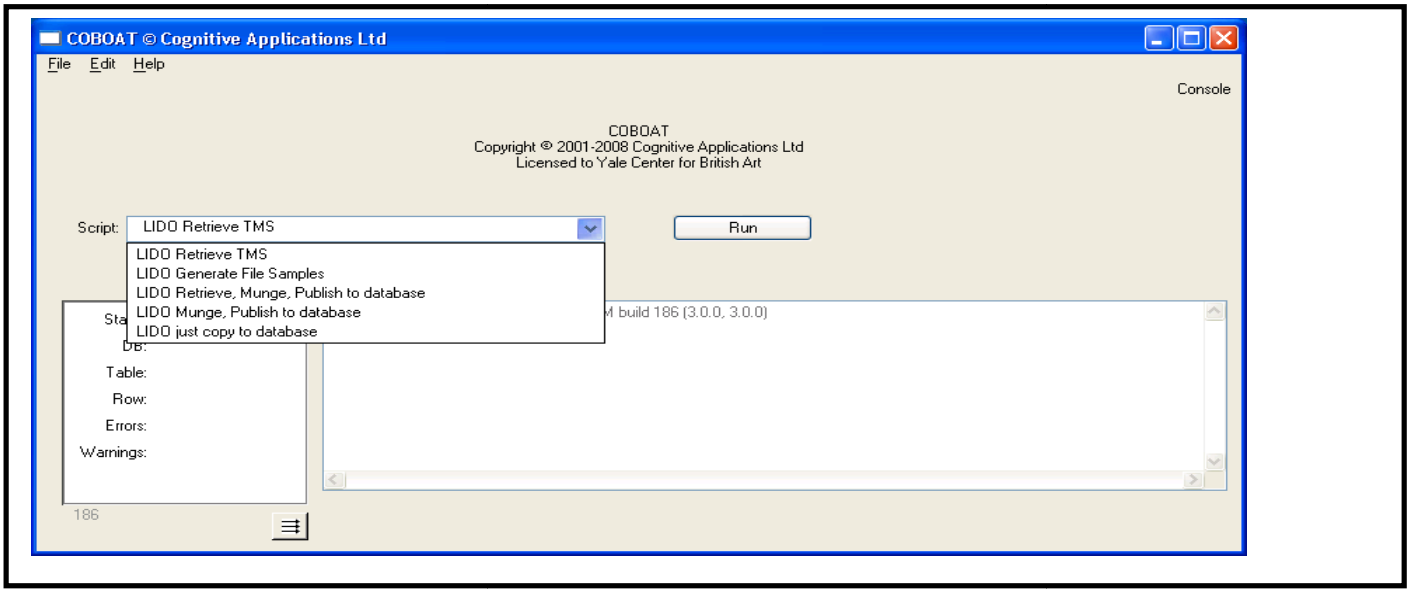
Using open source tools to expose cross collection data in the LIDO schema

A little more complicated than COBOAT, OAIcatmuseum requires support from the "open source" software Apache Tomcat and the MySQL database with the GUI interface Workbench. MySQL and Tomcat must be installed before OAIcatmuseum is installed. After Tomcat is running, the OAIcatmuseum war file is dropped in the Tomcat WebApps folder and expanded.



Use the open source MYSQL "Workbench" for a GUI interface to the OAIcatmuseum database. Workbench is similar to the SQL Server Management Studio.

Running COBOAT



A log file is created each time COBOAT is run. Run errors will be noted in the log file, but not always the cause of the error. It is important to frequently test each script during the coding phase to find errors. The COBOAT log file and the test xml documents and array files are good for debugging COBOAT errors.

Use MySQL workbench to review the xml documents and supporting tables in the OAIcatmuseum database.

Use Internet Explorer, Chrome or Firefox to review the xml documents from OAIcatmuseum.

Example of the URL used to review the xml documents residing in OAIcatmuseum;

http://bac4.yu.yale.edu:8080/oaicatmuseumlido/OAIHandler?verb=GetRecord&identifier=oai%3Atms.ycba.yale.edu%3A27601&metadataPrefix=lido

Questions, concerns or comments; please contact David Parsell at david.parsell@yale.edu