1. Introduction to the Yale Center for British Art

The Yale Center for British Art (YCBA) located at Yale University, positions itself as a public museum and a university research institute. Delmas-Glass (2012) found that “the YCBA holds the largest and most comprehensive collection of British art outside the United Kingdom, presenting the development of British art and culture from the Elizabethan period to the present day”. The YCBA collections consist of approximately 50,000 objects, 30,000 rare books and manuscripts and 30,000 reference library materials. Delmas-Glass (2012) stated that “the collections provide an exceptional resource for understanding British art”.

The YCBA art collections are cataloged in the Gallery Systems collections management system (TMS) and the YCBA library materials are cataloged in the Yale University Library ORBIS system.

The complexity of storing the YCBA collections in two different data bases is addressed by using the OAI-PMH protocol to harvest the YCBA collections for the cross collections repository and subsequently the YCBA searchable on-line collection. The data is harvested nightly, indexed and delivered through the Apache Solr search platform when queried from the Cross Collection Discovery (CCD) website or the searchable collection on the YCBA website.

The YCBA online collections catalog is sourced from the same database as the CCD, but the YCBA collections interface displays a richer data set than the CCD interface through the proprietary design focused on just the YCBA collections while the CCD interface is generalized to accommodate the diverse Yale collections. “The YCBA’s on-line collections catalogue (Delmas-Glass, 2012) is a single search box where, for the first time in the history of the museum, one can search across all the museum’s collections; works of art, rare books, manuscripts and reference library materials”.

By 2009 the YCBA had committed to a number of collection distribution objectives:

- Mounting a searchable on-line collection on the YCBA website;
- Contributing to the searchable on-line Yale database aggregating collection data from Yale cultural institutions; and
- Exposing the collection to harvesters outside the university community.

Yet the YCBA did not have the tools in place to meet these goals.

Empowered with the classic cultural institution mandate of great expectations but little manpower and funding, the YCBA sought open source solutions, eventually choosing tools available from OCLC Research (http://www.oclc.org). This decision, prompted by the low entry barrier and the necessity for an expedited solution, began an interesting journey that is still underway over two years after the YCBA first exposed the collection with OCLC tools.

Before we can discuss the data export software, we will review the collection cataloging procedures and the XML data schemas to establish the foundation for developing the project and the export tools.

2. Introduction to CDWA and CDWA Lite schema

As the standard data exchange schema in use at the beginning of the YCBA project, as well as the schema supported by the Museum Data Exchange tools, the CDWA selection was an easy choice for the YCBA. CDWA met the YCBA’s data export needs and provided a reasonably robust data schema as stated by Baca and Harpring (2009) “Categories for the Description of Works of Art (CDWA) describes the content of art databases by articulating a conceptual framework for describing and accessing information about works of art, architecture, other material culture, groups and collections of works, and related images”.
The CDWA document goes on to say that it includes 532 categories and subcategories and that a small subset of the categories are considered core data since they represent the minimum information necessary to identify and describe a work. The core data comprises the actual data schema called CDWA Lite and is best described by Baca and Harpring (2009) “CDWA Lite is an XML schema to describe core records for works of art and material culture based on the Categories for the Description of Works of Art (CDWA) and Cataloging Cultural Objects: A Guide to Describing Cultural Works and Their Images (CCO) (which uses a subset of the CDWA categories)”.

The detailed description of CDWA and CDWA Lite can be found at;

3. Introduction to the LIDO schema

The LIDO schema combines the CDWA and the European museumdat schemas. It is aligned with the SPECTRUM collections management standard and is an event-oriented approach compliant with the CIDOC CRM Conceptual Reference Model. The event-oriented approach of CIDOC-CRM allows generalizing the format of LIDO for use with extensive object data that identifies what events an object experienced in its lifetime, such as provenance, framing, history, publication, conservation history.

LIDO was introduced November 2010 and supersedes the CDWA and museumdat schemas.

LIDO embraces the same data sharing objectives as CDWA, but offers a much more robust schema to facilitate sharing much more complex event-driven data. “The strength of LIDO (ICOM-CIDOC Working Group Data Harvesting and Interchange, 2010) lies in its ability to support the full range of descriptive information about museum objects. It can be used for all kinds of objects, e.g. art, architecture, cultural history, history of technology, and natural history”.

LIDO is a flexible schema that is compliant with CRM, supports URL addressing and allows RDF modeling of collections data. The YCBA hopes that eventually linked data will solve a number of issues such as harmonized vocabularies and establishing a platform for British Art scholarship that uses the network environment as the database.

A detailed description of LIDO can be found at;
http://network.icom.museum/cidoc/working-groups/data-harvesting-and-interchange/what-is-lido/

4. Introduction to the Museum Data Exchange

The Museum Data Exchange Project (MDE) funded by The Andrew W. Mellon Foundation, empowered nine museums and OCLC Research to create tools for data sharing, analysis and aggregation. The project (Waibel, LeVan and Washburn, 2010) “established infrastructure for standards-based metadata exchange for the museum community and modeled data sharing behavior among participating institutions”.

The MDE project produced two tools; COBOAT, the main tool, extracts data out of collection management systems and generates xml documents in the CDWA Lite schema that are delivered to the OAICatMuseum data base on the OAI-PMH server. The default COBOAT configuration is set up for Gallery Systems TMS, but the configuration can be adjusted to work with other collection management systems. The peripheral tool, OAICatMuseum, sits on the OAI-PMH server data base and exposes the data to harvesters through the OAI-PMH protocol.

As stated by Waibel, LeVan and Washburn (2010) “the complete CDWA Lite data sharing strategy comprises:

- A data structure (CDWA) expressed in a data format (CDWA Lite XML)
- A data content standard (Cataloging Cultural Objects-CCO)
- A data transfer mechanism (Open Archives Initiative Protocol for Metadata Harvesting-OAI-PMH)

The MDE project objective is to lower the entry level barrier for data sharing by providing free tools to create and share CDWA Lite XML documents. This strategy is exactly what attracted the YCBA to the CDWA Lite schema and MDE. Essentially, with little expertise and investment, the YCBA could expose collection data for harvesting in the standard data exchange schema in use in 2009.

5. YCBA concerns regarding the Museum Data Exchange Tools.
Initially, the YCBA harbored reservations about using the OCLC tools. While the OAI-PMH protocol enjoys wide acceptance at cultural institutions, there had been little interest in COBOAT once the MDE project was completed. Also, concurrent with the 2010 YCBA efforts to expose data in the CDWA Lite schema, the cultural community was moving towards replacing CDWA Lite with the more robust LIDO schema, raising YCBA concerns that COBOAT might not be the long term solution it sought.
Fortunately, the YCBA made an excellent decision. COBOAT turned out to be a very efficient and flexible tool that easily managed exposing data in the CDWA Lite schema and allowed the YCBA to extend COBOAT far beyond its original capabilities in order to support the LIDO schema.
The beauty of open source software is building on existing tools to produce a solution that meets your own needs and can be shared with the community for the greater good. This is exactly what the YBCA did by extending the MDE tools to support the LIDO schema after LIDO superseded the CDWA Lite schema; essentially producing 2nd generation MDE project tools.

6. Getting the YCBA project started
In 2009, the YCBA wished to expose collection data in the CDWA Lite schema, the accepted standard in use at the time. Optimally, the YCBA prefers to use open source tools, and to that purpose turned to the OCLC Mellon-funded Museum Data Exchange project that had built an open source data export tool based on the proprietary COBOAT software. The OCLC project focused on exporting data from the Gallery Systems collection management system called TMS, but the software is adaptable to other collection management systems.

A caveat in the YCBA “open source” software strategy is the fact that COBOAT is proprietary software as described in the Museum Data Exchange Project – Installation and Configuration Manual v.009.

“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”. (Rubinstein, Rudge and Norris, 2008)

Fortunately, the MDE project ensured that COBOAT would be free to all cultural institutions wishing to use the MDE tools. The only requirements are that cultural institutions must request a license by email from Cogapp, and COBOAT needs the license file installed in a sub-folder to be fully functional.

When the YCBA extended COBOAT to support the LIDO schema, COBOAT drifted outside the terms of the original Cogapp MDE license. However, when requested by the YCBA, Cogapp generously agreed to extend the free MDE COBOAT license to include exporting collection data in the LIDO schema.

The YCBA began using the MDE tools in 2010, with the objective of exposing collection data in the CDWA Lite schema, but when the more robust LIDO schema superseded CDWA Lite, the YCBA reengineered the project to focus on the LIDO schema.

Without open source tools available for exposing data in the LIDO schema, and the positive 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 successful extension to the LIDO schema.
In 2011, the YCBA extended COBOAT and OAICatMuseum to allow generating LIDO schema XML records and exposing them for harvesting using the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). Both the CDWA Lite and the LIDO schema versions of COBOAT and OAICatMuseum are “open source” software available on the OCLC website. COBOAT requires a free license obtained by emailing Ben Rubenstein at Cogapp benr@cogapp.com.

Once the MDE tools had been extended to expose data in the LIDO schema, Cogapp extended the COBOAT license to include both the CDWA Lite and the LIDO schemas.


7. YCBA checklist for cataloging and exposing collection data with COBOAT and OAICatMuseum

Following are the areas that need to be addressed when planning to expose collection data.

1. Identify and document project objectives.
2. Review and document website data presentation objectives to identify cataloging needs.
3. Review collection data to determine what cataloging needs to be done.
4. Initiate cataloging projects to prepare the collection for export and website display.
5. Create a document linking collection data fields to the XML document fields.
6. Create a sample XML document, and using a tool such as Oxygen, validate the XML schema.
7. Install and test the tools to be used for data export and exposure for harvesting.
8. Write and test the needed scripts.
9. Document the installation and modification to the tools.
10. Extensively test the system before exposing production collection data.

8. YCBA project objectives

By 2009, the YCBA had committed to three collection distribution objectives, but did not have the tools in place to meet the commitments, which are

1. Building on-line collection search functionality on the YCBA website.
2. Exposing the collection to a Yale University central searchable data base fed by all of Yale’s data providers, which includes three museums, the library system and others. Yale University sought to build a single-search portal for Yale University collections.
3. Exposing the collection for harvesting by aggregators outside of Yale.

To expedite a solution to meet the three objectives, the YCBA focused on exposing the collection data for harvesting by the Yale University cross collection database (CCD). Once the harvesting software was in place, the YCBA on-line searchable collection would be designed to retrieve data from the CCD. Eventually, exposing the data for harvesting would be expanded to include harvesters outside Yale University. Figure 1 below illustrates the Yale Cross Collection database harvesting design.

9. Cataloging the collection database prior to exposing the collection

Before a cultural institution can begin exposing data, there has to be sufficient collection data to expose, which is not as simple as it sounds. Basic tombstone data in a collection management system is not enough for an interesting website and does not present well in a searchable on-line collection without robust supporting data to
enhance the tombstone data and facilitate searching.

The first step to cataloging the collection for export is identifying what data you want to expose and how you want to present it (in this case, a searchable collection on a website). Once the goals have been determined, cataloging projects can be implemented to catalog the appropriate data.

As stated by Delmas-Glass (2011) “cataloging guidelines need to be set for creating sharable metadata, which is Human understandable and can be processed by a Machine, is useful and usable to services outside of its local context and is quality metadata”.

For the YCBA, creating shareable metadata meant defining metadata standards, the metadata schema, the data content standard, and the data content values (controlled vocabularies).

Since the CDWA was the standard schema and supported by the MDE tools, the YCBA selected a core subset of CDWA as the metadata schema, and the Cataloging Cultural Objects standard (CCO) for the data content standard. The CCO lists clear guidelines for selecting, ordering, and formatting data values as stated by Baca, Harpring, Lanzi, McRae, Whiteside (2006) “Cataloging Cultural Objects is a system of metadata standards. It facilitates best practices that are fundamental to enhanced online access and retrieval”.

The benefits of the YCBA application of the standards was described by Delmas-Glass (2011) “designing the collection database to be standard compliant allowed the YCBA to make use of the whole suite of standards for cultural institutions; the CDWA Lite schema and the OAI-PMH protocol”.

Additionally, the YCBA identified data facets to display in its searchable collection (a step towards thinking about data as data rather than data as display). Initially, the YCBA was limited by the CDWA Lite schema, but still found interesting intersections between the art collection metadata and the library metadata for the facets. On a cautionary note, the multiple data layers generated by the facets required the YCBA to carefully select the fields indexed as facets.

10. YCBA cataloging projects in preparation for exposing the collection data

Constituents
1. Alternate names
2. Text entries (useful fields for including additional data that does not fit into the standard database fields)
3. Life roles
4. Place of activity (geographic locations with GIS data)
5. Keywords linked to the AAT, ULAN, TGN

Objects
1. Work type
2. Dimensions
3. Copyright
4. Text entries (useful fields for including additional data that does not fit into the standard database fields)
5. Provenance
6. Linked images
7. Geographic locations with GIS data
8. Keywords linked to the AAT, ULAN, TGN

Other
1. Bibliographic data
2. Exhibition data

11. Linking the YCBA collection to the XML data schemas

In conjunction with designing the cataloging projects, the linking between the collection database fields and the XML export fields is documented in a detailed mapping document and a model XML document. Figure 2 below is the mapping document linking the collection database fields to the CDWA Lite and LIDO schema fields.
Figure 3 below is the sample LIDO schema XML document verified in Oxygen, the only software that the YCBA purchased for this project. Oxygen is a valuable tool for verifying your XML schema against the appropriate schema website.

The YCBA is committed to using open source tools that would allow the YCBA to quickly implement systems to meet the three objectives. The OCLC Museum Data Exchange project was selected for its low entrance requirements and the possibility to be the one system to meet the three YCBA objectives. The MDE is comprised of two tools, COBOAT to export the data from the collection database and OAICatMuseum to expose the data for harvesting.

Figure 4 above is an illustration of the MDE tools and the supporting software required to expose collection data. The above diagram illustrates the data flow from the collection database to exposure for harvesting.

**12. COBOAT and OAICatMuseum need the following open source software to support the data export and harvesting.** All of them can be downloaded from the web.

1. MySQL database software. COBOAT sends the XML documents to the OAICatMuseum db for harvesting.
2. MySQL workbench; a GUI interface which expedites reviewing the XML documents in MySQL.
3. Apache Tomcat to enable harvesting from the OAICatMuseum db.
4. A good text editor for writing the COBOAT scripts. Notepad++ was used for this project.
5. Software to validate the XML schema. Oxygen was used, which must be purchased.
COBOAT and OAICatMuseum are downloaded from the OCLC.org website. The downloads include comprehensive manuals providing excellent guidance for the installation, coding and operation of the software.

COBOAT and OAICatMuseum together perform only one task; extracting data from a collection management system, organizing the data in XML documents and exposing them 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 the MySQL OAICatMuseum database where the XML records are exposed to harvesters.

The Installation and configuration manual states (Rubinstein, Rudge, Norris, 2008), “COBOAT might best be summed up as a scriptable tool for the automation of data migration and conversion tasks”.

That is one of the beauties of COBOAT; the use of scripts to allow quick changes and re-testing instead of the much more time-consuming procedures required with fully compiled software.

The COBOAT manual *Museum Data Exchange Project – Installation and Configuration Manual v.009*; states that “the concept for COBOAT is to centralize all the code that knows how to retrieve data from or build data to various databases, scan images etc; configured by simple scripts which as much as possible contain only the details specific to a given installation; and which where possible hide the details of individual database systems or connections, so that if these change, only minimal alterations are required to the scripts”. (Rubinstein, Rudge, Norris, 2008)

COBOAT works on either the Mac or Windows platform and uses a small footprint single folder with all of the supporting files and configuration scripts residing in sub-folders.

The YCBA COBOAT and OAICatMuseum implementation uses two servers as diagramed below in Figure 5

- Database server for the collection.
- OAI-PMH harvesting server exposing the collection.

13. YCBA Implementation of COBOAT and OAICatMuseum

COBOAT and OAICatMuseum can be set up in a number of different configurations. The 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 second dedicated harvesting server.

Following are other possible server/workstation configurations, each with performance ramifications.
1. COBOAT and OAICatMuseum resident on the collection database server, which compromises your collection database when you expose the server to harvesters.
2. COBOAT installed on a workstation with the collection database on a server. This is the slowest configuration based on the network traffic generated and the lower powered workstation. Also, it requires a workstation with considerable hard drive space to store the exported data and XML documents.
3. COBOAT and OAICatMuseum installed on a workstation. This is a bare bones low cost solution that will be very slow due to the network traffic between the workstation and the database, and the limited workstation power. This configuration requires even more hard drive space to accommodate the data export and the second copy of the XML documents in the MySQL database.
14. Detailed technical guide to installing and scripting COBOAT and OAICatMuseum

Figure 6 above illustrates the main COBOAT folder and the sub-folders.

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 each perform a specialized task.
1. The TMSretrieve script exports data from the CMS and stores it in text files in preparation for building the XML
documents. I chose to write the export queries in Transact-SQL, but you have the option of using other languages
such as $smarty and php. I put all pre-processing logic in this script for simplicity.

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.

Drilling down in the Service folder in figure 7 below reveals the subfolders where all of the processed data is
stored.
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. The fields 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 contains a copy of the XML documents sent to the main MySQL table, as well as the supporting data stored in the other MySQL tables in the OAICatMuseum database. 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 “TMSretreive” 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.

15. Figure 8 below is a detailed look at the source data folder and the data files extracted from the collection database.

16. Extending the CDWA Lite schema to 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. 300mb.
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.
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 Scheduler”.
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 connects to the TMS database with an ODBC connector. “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.

When programmatically probing a SQL server database, 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. Remember, SQL server will only allow a SQL login to programmatically probe SQL server.
On the YCBA DEV server installation, the following SQL script is run daily by the SQL agent to reset the COBOAT SQL login password. Replace ‘SQLlogin’ with your SQL login ID and ‘password’ with your designated 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 set of 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.

17. 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 XML document. If the checksums are the same for the same objectID, COBOAT sends the stored older 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.
When testing, there are two workarounds to guarantee sending all of the XML documents to the OAICatMuseum db. One is to delete the checksum history file named cdwaliteCache.txt in the METADATA folder. The second is to delete the OCM_metadata_record.txt file in the Outputdb folder. Both will force COBOAT to rebuild the file and send all of the new XML documents to the OAICatMuseum db.

18. Deleting released records from external databases
The delete 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. Working with the YCBA, 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 to ensure deleting the records from all external databases.

19. 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.

During the CDWA Lite phase of the project, 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 = 7 – 9 kb. The average LIDO document size = 26 – 38 kb. The much larger LIDO XML documents forced changing the processing parameters from the default; chunkrows="100" chunkinput="10000"
to;
chunkrows="10" chunkinput="10". On the production TMS server, no speed difference between using 10 or 100, but the smaller dev server only works when set to 10.

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 approx. 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. 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"

Unlimited size text fields are a problem for COBOAT, which could not process them during the TMSretreive function. The problem was solved by using a field type conversion in the TMSretreive script. Following is an example of the conversion code;

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

The large field size specified in the conversion is not a problem as long as the field is defined with a set size.

20. TMS 2010 upgrade to TMS 2012 with Unicode fields

Fall 2012, TMS was upgraded to the 2012 version which converted all of the database text and varchar fields to nvarchar Unicode fields. While beneficial for TMS, the change caused severe problems for COBOAT, which could not correctly export Unicode data from SQL server. Instead of the expected data in text and varchar fields, only the first byte of each field’s data was exported.

Searching on the web revealed this to be a known problem with SQL server 2008. SQL server stores the Unicode data as UTF-16. SQL server 2008 doesn’t have the correct collation to convert the Unicode fields to text when
exporting SQL Unicode data to txt files. The only solution found so far is to convert every text and nvarchar field in the TMSretrieve script that extracts the collection data from the SQL database and saves it in txt files. Following is a code sample. The varchar size can be less than 8000, which is the maximum size allowed.

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

Although a tedious solution, it does work correctly and was implemented by YCBA prior to the TMS 2012 upgrade.

Additional problems from the TMS 2012 upgrade is the reduced TMS response to user activities. Increasing the production server memory from 4 to 8gb has helped considerably. Queries and reports are being reviewed to identify ones that will benefit from rewriting to increase response times.

21. OAICatMuseum implementation

A little more complicated than COBOAT, OAICatMuseum requires the installation of and 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 as displayed in figure 9 below.

Use the open source MYSQL “Workbench” for a GUI interface to the OAICatMuseum database. Workbench is similar to the SQL Server Management Studio interface.

The OAI-PMH website [http://www.openarchives.org/pmh/](http://www.openarchives.org/pmh/) (2/27/13) best describes;”The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a low-barrier mechanism for repository interoperability. Data Providers are repositories that expose structured metadata via OAI-PMH. Service Providers then make OAI-PMH service requests to harvest that metadata. OAI-PMH is a set of six verbs or services that are invoked within HTTP”.

22. Running COBOAT,

The COBOAT executable is CB186.exe and is located in the COBOAT directory. The COBOAT menu displayed in figure 10 above is built from user scripts that call the various COBOAT modules. The scripts and the COBOAT menu are easily customized. The COBOAT executable can be run manually or automatically with the Windows Task Scheduler. 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. Copies of the log file can be found in the COBOAT folder and a few of the data folders under the Service folder. 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.
You can harvest one or a series of XML documents using the OAI-PMH protocol. Following is the URL for harvesting one XML document where the ObjectID = 4908. This URL is specific to the YCBA harvesting server setup and should be all one line in your browser without spaces.

http://bac6-dev.its.yale.edu/oaicatmuseum/OAIHandler?verb=GetRecord&identifier=oai:tms.ycba.yale.edu:4908&metadataPrefix=lido

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

COBOAT can be found at http://www.oclc.org/research/activities/coboat.html

OAICatMuseum can be found at http://www.oclc.org/research/activities/oaicatmuseum.html

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

References


• Delmas-Glass, Emmanuelle. Metadata Strategies for a Cross Collection On-line Catalogue. 2011

• Delmas-Glass, Emmanuelle. Using open source tools to expose cross collection data in the LIDO schema. 2012

• LeVan, Ralph, Gunter Waibel and Bruce Washburn. Museum Data Exchange: Learning how to Share. 2010

