Digital Asset Management and Publication with LadyBird

Eric James, Lakeisha Robinson, Osman Din, Jay Terray, Rebekah Irwin, Mike Friscia

Conference Submission to Open Repositories 2013

*Yale University Libraries*

[eric.james@yale.edu](mailto:eric.james@yale.edu)

LadyBird is an application designed to provide a consolidated solution for the management of metadata and digital assets at Yale University Library. It provides a configurable system for migrating digital objects and collections, normalizing metadata, and preserving and publishing content. It was initially writing in Microsoft .Net and C#, hosted on Windows 2008 using Microsoft SQL Server 2008. It is currently being migrated to Java and Ruby. This paper will describe how Ladybird handles the workflow from submission of content to publication.

To accommodate varying needs, Ladybird establishes a hierarchy in terms of 1) collections which define content owners across campus, and 2) projects which can be owned by a collection. At the heart of Ladybird is the object, belonging to a collection and its project. At the project level workflow and metadata description is defined. Objects can be designed to be atomic (such as an image), or complex (such as a book with pages) A typical workflow involves 1) adding digital files to a job folder created within a project, 2) creating a spreadsheet describing the assets as individual objects, 3) processing these into the Ladybird database, 4) publishing these to access and preservation systems such as Fedora.

The primary association of metadata with data (content) occurs in the excel spreadsheet submitted for ingest. The rows of the excel doc are individual objects identified by an oid (object id) in the first column. In the columns of the spreadsheet are metadata fields identified by fdid (field definition id), and other special processing fields. The excel fields can be pre-populated from existing objects, manually entered by the cataloger directly to the excel doc, or manually entered through the LadyBird GUI. The oid column can be prepopulated if existing or empty if new. The excel document allows for the creation, update or deletion of records, and for cataloging and processing in batch. Metadata can also be ingested from existing metadata in the MARC,EAD, and DC formats as identified from their respective identifiers in Orbis (the Yale ILS), and YFAD (the Yale Finding Aid Database).

As mentioned in the previous paragraph, each column in the excel doc is defined by a fdid. A table in the database relates the fdid to a field definition, in function creating a pool of metadata fields with which to describe each object. For each field definition of an object (a cell in the database) the value from the spreadsheet is kept in the tables either as a string, a longstring, or acid (authority control id) value. Acid values are numbers corresponding to a controlled vocabulary maintained by the database, a particularly powerful feature. Depending on the project different field definitions can be invoked. In addition to the metadata field definitions, other fdids (columns) are saved for non-metadata processing purposes, such as identifying the path to the file, or signaling for the deletion of a record. When processing is triggered, either through the GUI or in batch, the excel doc is parsed and stored in a set of ingest tables. Other non Excel processing may occur at this time converting EAD, MARC and DC into field definitions in the ingest tables. The tracking of the entire process is also maintained in its own set of tables. After ingest, either from the excel doc or the metadata schema files, the data and metadata is converted into the data tables. Each collection has its own set of data tables what are known as the c# (“c number”) tables, the number corresponding to the collection id. If this is an update process, the current c# table records are moved to the the p# (“p number”) table and replaced. This system provides robust audit and update at the collection level.

At this point a representation of the data and metadata are secure in the system. The LadyBird GUI offers search functionality to retrieve images and metadata mainly for cataloging and QC purposes. The final step is to export/publish this data and metadata. Ladybird is designed to provide pluggable export functionality to Excel, a zip file, XML, BagIt, ContentDM, Artesia, and Fedora. This paper will describe the Fedora process.

Objects flagged for publication to Fedora are entered into a table named hydra\_publish, which serves as a tracking queue for fedora ingest. Each row represents an oid. There are fields defining tracking timestamps, create/update/delete functionality, parent oids for complex objects, and contentModel corresponding to the Fedora Content Model.

An application currently being written within a ruby/rails/hydra project, uses the hydra\_publish table to manage the Fedora ingest. The ingest script queries for the first top level (parent complex object or simple object) in the table ordered by timestamp and processes it according to the content model and repeats this polling continuously. There can be any number of content models, as of now 2 are implemented, “simple” and “compound”. A “simple” object refers to a tif to be converted into an atomic object in fedora. One of the columns in the import excel doc specifies the path of a tif, either uploaded to LadyBird in a drop folder, or existing in permanent attached storage. LadyBird then creates derivative jp2 and jpg files, transfers these files to permanent storage as necessary and creates a MODS, a hydra rights file, and an access conditions file as metadata. The metadata is serialized into these file formats from the field definition values in the object’s c# tables. For each object in the hydra\_publish table there are rows in a hydra\_publish\_paths table for the tif,jpg,jp2,MODS,rights, and access with all the properties necessary for Fedora ingest. The object is then ingested using ActiveFedora within a Hydra task. Complex objects work similarly. One top level object is found in the hydra\_publish table and ingested, and then its children are ingested one by one. Once in Fedora and indexed by Solr, the content is then available for publication through blacklight.

LadyBird is a work is a work in progress. There are plans to integrate with an Archivematica solution for use of their preservation and publication services within the next year. There is an estimated 4 year backlog of content to submit through this workflow including ~300k images and 8 collections of varying sizes of 50k to 2M objects. It is planned that the system will scale to accommodate other content models, including AV and maps, and be retrofitted entirely as a hydra head. All in all, the LadyBird system, with its metadata format agnostic normalization, controlled vocabulary capability, built in update/delete/auditing, and its potential for pluggable publication to Fedora and other systems, provides a foundation for use and reuse that should fit the needs of the library for digital assets and metadata processing for years to come.

1 <http://ladybird.library.yale.edu/>

2 <https://www.archivematica.org/>