SCHOLARLY PUBLISHING OFFICE WHITEPAPER*

Plan for implementing a uniform content ingestion system for SPO

Kevin S. Hawkins

Summary

This paper charts a course for development of a uniform content ingestion system for SPO, building on plans documented in two previous whitepapers: *Proposals for a New Workflow for Level-4 Content* and *Plan for Implementing a Web-Based Unified Bibliographic Database for SPO Publications*.

SPO needs to develop a uniform content ingestion system in order to establish a more scalable workflow and enforce uniform metadata standards. This system needs to have four components:

- A unified bibliographic database (UBD) for headers of Text Class items and possibly for Bib Class records
- A script for exporting records from the UBD for use by prep scripts.
- A web interface to the UBD with access controls that allow authorized users to edit and submit records for certain collections, plus submit "bodies" for Text Class records.

Motivations

There are a number of disadvantages to the current system of having separate FileMaker databases for each publication:

- There is no way to enforce standard encoding across collections.
- The use of field calculations predates XML support in FileMaker, so the output is only checked for validity when running a prep script. (A more XML-centric workflow would have SPO staff export FileMaker databases as XML and then transform them with XSL.)
- Access to these databases can only be provided by DSS hosting them. Security issues make remote access to both version 6 and 7 databases difficult, so Panagis would prefer that we not use FileMaker at all.
- Inputting non-ASCII characters presents problems: FileMaker won't store a character for which a glyph is not available in the font being used in the FileMaker form. This will continue to be a problem as long as there are no fonts with the same name available for Windows and Mac OS X that have all non-ASCII glyphs needed. (A lack of appropriate fonts available for Windows and Mac OS X would also be a problem in any other solution, but at least most programs and web interfaces are capable of storing a character even if it can't display it.)

Unified Bibliographic Database (UBD)

The UBD should hold metadata for all items in SPO's Text Class collections. It should possibly hold all Bib Class records as well, though the future of SPO's support for Bib Class collections is unclear pending discussions with Library Systems, which plans to hire a new staff

member to support bibliographies. A relational model needs to be developed that is sufficiently normalized for current needs but reflects a data model abstract that it will be usable in the future; in particular, the relations between items, DLPS IDs, collections, DocEncodingType, and DLXS Class need to be modeled.

The UBD structure should be abstracted from DLXS encoding as much as possible, with transformation to appropriate tagging (based on the collection type) done by the export script. This will ensure maximum portability of the metadata to other delivery systems and, even if DLXS continues to be used, allow for the creation of archival records in another format besides Text Class and Bib Class.

IDNOs for records will be calculated based on metadata entered; uniqueness of IDNOs in the entire database will be enforced. For other fields, controlled vocabularies will be used when applicable.

In the first phase of implementation, the UBD will replace individual FileMaker header databases for `serialarticle` and `serialissue` collections. (The metadata ontology of these two collection types is practically identical, though they have different export formats.) In the second phase, it will replace individual `.hdr` files stored in the prep directories of `monograph` collections. In the possible third phase, it will replace individual FileMaker header databases for Bib Class collections.

UBD to DLXS export script

We should have a command-line script which can be called from the collection prepping scripts (`makepub.sh` or `makebackfile.sh`). This script should be in CVS at `bin/s/spocolls/UBD2DLXS.pl`. Given a collid, it will export all records in this collection from the UBD and transform using XSL to header or record format for that collection type (`serialarticle`, `serialissue`, Bib Class, or `monograph`), outputting `prep/c/collid/headers/collid_headers.txt` for all Text Class collections and `prep/c/collid/articledivs/collid_articledivs.txt` for all `serialissue` collections. The output location of Bib Class records needs to be determined in consultation with Jeremy.

We will need to be able to write custom export scripts, such as one for generating the article list in the Diderot project.

UBD interface

We need a web interface for both SPO staff and publishing partners to edit and submit records, plus submit “bodies” for Text Class records. It should probably be located at `web/s/spo/ubd/` .

All users inside the DLPS/SPO IP range should have access to all functionality of the web interface. All full-time SPO staff should be able to authenticate from other locations for full access privileges. All SPO publishing partners should have full access to records for only their collections. This access will likely be managed using the Library auth system, but U-M’s LDAP service is another option.

Users entitled to access records for more than one collid should be prompted to choose which collid's records to view, though there should be an option to view records from all authorized collections.

For authorized collections, users should be able to filter records viewed by any field value and create, modify, and delete records one at a time.

The field for the record's IDNO should not be editable since it is calculated from other values. All other fields should be editable, though the user should be prevented from choosing values that would cause an IDNO conflict.

When creating records for a journal, the interface should provide a controlled vocabulary of journal titles (corresponding to DLPS IDs) in that collection. For most collections, there will be only one title, so all records will have that title by default. For journal collections, in addition to being able to create a blank new record, users should also be able to create a new record for an article in that issue: this function would duplicate field values in common for all articles in an issue, plus automatically increment the article number.

The interface should give guidance to users entering metadata, with notes explaining metadata conventions (similar to text that currently appears in red in FileMaker databases). There should be notes used for all collections, notes for specific collection types, and collection-specific notes, and journal-specific notes when applicable. These might be stored in configuration files similar to those used by the DLPS stats system, stored in CVS and modifiable by those with accounts on a DLPS/SPO server. These configuration files might also specify which fields are required and might specify any required datatype or list of acceptable values.

Users should be able to upload three types of files for Text Class records (both new and existing) through the interface:

1. **the "body" file**: This would be stored at `prep/c/collid/inbox/`, automatically named to match the IDNO of the associated record, for further processing by SPO staff. One body is required for each item, but there should be an option for more than one body per IDNO to be uploaded; subsequent files would have `a`, `b`, `c`, etc. appended to the end of the IDNO.
2. **"inline images"**: These would be stored at `img/c/collid/` and named according to standard inline image naming scheme, numbered in the order they are uploaded. No inline images are required for an item, but there should be an option for more than one per IDNO.
3. **PDF version**: Users not uploading a PDF version should be reminded that they haven't done so when creating a new record. If uploaded, this file would be stored at `prep/c/collid/pdfs/`, automatically named to match the IDNO of the associated record.

All files uploaded should have 0664 permissions.

Finally, the interface should have a report generation option, which displays a short report of modified or created since the last time the report was generated, and files added since then. This report can then be emailed to an arbitrary email address. Content providers will be encouraged to use this to send a message to their contact at SPO to notify SPO that new content is ready to be processed. This is useful less as a notification tool and more as an inventory of newly created or modified content, which is helpful for SPO, especially when tracking inline images that need to be inserted into the markup by hand.

Typical use

The following is the envisioned workflow for publishing a new issue of a journal:

1. The publishing partner creates records for articles in the new issue and uploads files for these articles. When finished, the publishing partner generates a report and notifies his or her contact at SPO.
2. The SPO staff member copies the files from `prep/c/collid/inbox/` to their computer for further prep work, including extracting text, applying `spostyles`, or both. When finished, the SPO staff member uploads the prepped content to the appropriate directory or directories in `prep/c/collid/`.
3. The SPO staff member runs `makepub.sh` or `makebackfile.sh`, which in turns runs `UBD2DLXS.pl`, automatically exporting the collid's records from the UBD.
4. Further prepping continues as in current procedures. After prepping, SPO staff moves all the files from `prep/c/collid/inbox/` to `prep/c/collid/inbox/previous/`.

Plan for implementation

I suggest the following plan for implementation:

Phase I: serial publications

1. Seth writes provisional database models for the UBD, keeping serial publications in mind, with different levels of normalization and abstraction.
2. Seth writes a standard XML format for exporting the data. At this point it will likely be helpful to consult a list of header labels that have been customized at the collection level in DLXS. Jeremy can generate this for Seth.
3. Seth, Jeremy, and Kevin examine this model for sound design and the XML format for compatibility with DLXS.
4. Kevin creates a new “export” field called `UBD2DLXS` in existing FileMaker header databases to match the standard XML format. If changes to the provisional database model are needed, he consults with Seth and Jeremy.
5. For any collections for which `UBD2DLXS` does not match the old “export” field, Kevin tests exporting, prepping, and reindexing collections using `UBD2DLXS`, comparing filtering of headers and reslist details to ensure continuity with the old “export” fields.
6. Seth sets up the UBD database on the [development/production?] MySQL server.
7. Seth writes `UBD2DLXS.pl`.
8. Data is imported from the `UBD2DLXS` field in existing collection databases into the UBD for testing `UBD2DLXS.pl`: verify that `collid_headers.txt` generated using `UBD2DLXS.pl` is identical to `collid_headers.txt` generated using the `UBD2DLXS` field in FileMaker.
9. Seth develops the UBD interface.
10. SPO staff and students test the interface and learn about its envisioned role in the prepping process for serials.
11. One collection at a time, data is imported from existing collection databases into the UBD, and the corresponding FileMaker header database is decommissioned. Publishing partners are taught to use the UBD to submit content and metadata to SPO.

Phase II: monographs

12. Seth examines the database model to see if monograph headers can be fit into it, proposing changes needed to accommodate them.
13. Seth modifies `UBD2DLXS.pl` as needed to accommodate monograph header records.
14. Data is imported from `.hdr` files into the UBD for testing `UBD2DLXS.pl`: verify that data exported using `UBD2DLXS.pl` is identical to `.hdr` files.
15. Seth modifies the UBD interface as needed to accommodate monograph records.
16. One collection at a time, data is imported from the `.hdr` files into the UBD.
17. When all collections have been imported, change `makepub.sh` to require a `collid_headers.txt` file to be present and to delete `.hdr` files after prepping. SPO staff are taught to use the UBD for these collections.

Phase III: bibliographies (assuming SPO continues to support Bib Class collections)

18. Seth examines the database model to see how Bib Class items can be fit into it, proposing changes needed to accommodate Bib Class records.
19. Kevin creates a new "export" field called `UBD2DLXS` in existing FileMaker header databases to match the standard XML format. If changes to the database model are discovered at this point, he consults with Seth and Jeremy.
20. For any collections for which `UBD2DLXS` does not match the old "export" field, Kevin and Jeremy test exporting, prepping, and reindexing collections using `UBD2DLXS`, comparing interface display of records search and browsed to ensure continuity with the old "export" fields.
21. Seth modifies `UBD2DLXS.pl` as needed to accommodate Bib Class records.
22. Data is imported from existing collection databases into the UBD for testing `UBD2DLXS.pl`: verify that data exported using `UBD2DLXS.pl` is identical to data exported using the `UBD2DLXS` field in FileMaker.
23. Seth modifies the UBD interface as needed to accommodate Bib Class records.
24. One collection at a time, data is imported from existing collection databases into the UBD, and the corresponding FileMaker header database is decommissioned. Publishing partners are taught to use the UBD to submit content and metadata to SPO.

Future development

Much of this proposed system reflects the needs of working with DLXS and particulars of DLXS's architecture. However, the system could be modified to work with another delivery system, and it could integrate additional future components.

Divorcing from DLXS

The ingestion system could work with a delivery system besides DLXS by changing the following:

- The script `UBD2DLXS.pl` would be replaced by another to export the metadata needed by another delivery system or to an archival format besides Text Class or Bib Class.
- Guidance offered to users of the interface would be changed to reflect the realities of a different delivery system, which might not force the same metadata decisions currently required by SPO's use of DLXS.

- "Bodies" submitted might be deposited in another location besides `prep/c/collid/inbox/`, perhaps feeding directly into the delivery system rather than waiting for additional prepping.

Peer-review management system

If SPO implements a peer-review management system, we should consider whether there should be an automated way of transferring data from one system to the other. Avoiding manual work is always desirable, except when a manual step gives an opportunity for error-checking. Articles under peer review may have their titles, authors, keywords, and even volume and issue numbers changed, so there is value in verifying all metadata at the end of the peer-review process. For this reason, it may be useful to force users to take all of their content out of the peer-review management system and put it into the ingestion system by hand rather than allowing them to transfer it automatically. Further exploration of peer-review management software will clarify the best option.