Automated Workflow for the Ingest and Preservation of Electronic Journals

Evan Owens; Portico; Princeton, New Jersey, USA

Abstract

Portico (http://www.portico.org) has developed an automated workflow for ingest of publisher-supplied e-journal source files into a preservation repository. The components of the workflow include package disassembly, format identification and verification, structure mapping, automated metadata harvesting, rule-based format normalization, and support for quality control and inspection.

Introduction

Portico is a new, not-for-profit electronic archiving service established in response to the library community's need for a robust, reliable means to preserve electronic scholarly journals. Portico was initiated by JSTOR and has been developed with the initial support of Ithaka, The Andrew W. Mellon Foundation, and the Library of Congress. Portico's mission is to preserve scholarly literature published in electronic form and to ensure that these materials remain accessible to future scholars, researchers, and students.

Portico takes a format-based migration approach to the long-term preservation of electronic journals. Publishers supply their electronic materials such as SGML or XML texts, page images, graphics at various resolutions, media, data and other files for archiving. These are the source materials for their print and web products rather than the web rendition files and are thus less dependent on current rendition technology. Portico normalizes proprietary publisher XML and SGML DTD instances to the NLM Archiving and Interchange DTD so as facilitate future rendition and to reduce the number of formats for preservation and future management. As part of that normalization, publisher-specific business practices such as file naming conventions are resolved and eliminated.

The technological roots of Portico are found in the work done in the Andrew W. Mellon Foundation's E-Journal Archiving Program (1999) [1]; in particular, the work of Harvard University Library on an archival DTD [2] which in turn influenced the National Library of Medicine's Archiving and Interchange DTD [3]. Also important in the design of the Portico workflow was the organizing meetings for a Global Digital Format Registry (GDFR) in 2002-2003 [4] and participation in the RLG/OCLC PREMIS Working Group (Preservation Metadata Implementation Strategies) in 2004-2005 [5]. Key technologies used include XML, XML schema, Schematron [6] for XML semantic validation, NOID [7] for accession IDs, METS [8], and JHOVE [9] for format verification, as well as Documentum, Oracle, JMS, and LDAP.

System Overview

The Portico system is a set of applications, as shown in Figure 1. Before publisher content can be ingested into the archive, samples are received, analyzed, and any necessary publisher-specific profiles or tools are developed, tested, and deployed. That is represented by top row of boxes in Figure 1. The middle row of boxes is the main operating systems which are the focus of this discussion: the Content Preparation System and the Archive Management System. The Portico Content Preparation application consists of a Documentum repository and workflow, a java application that controls the automated process described below, and a set of registries that contain information about formats, tools, and other enterprise-level information. The system is described in more detail elsewhere [10].

The Portico Archive is, strictly speaking, dark: all content is visible only through a separate delivery application run on our behalf by our sister organization, JSTOR [11]; this leverages JSTOR's experience with public web sites and delivery infrastructure and allows Portico to concentrate on archiving and content management.

Process Overview

At the highest level all data processing workflows look pretty much the same:

- Content receipt
- Batch creation
- Automated processing
- Quality control and problem resolution
- Content release

In our process all these steps are essentially a "pre-processor" that leads to archival ingest and are part of the Content Preparation System. The inputs to content preparation workflow are arbitrary, publisher-specific collections of data with proprietary file and directory naming conventions often minimally documented and sometimes inconsistently applied. The output of the workflow is normalized content with complete descriptive, technical, and events metadata packaged in Portico METS files ready for ingest into the archive.

The noteworthy aspects of the workflow are all contained in the automated processing phase shown in Figure 2. Some detail is omitted for clarity; the actual implementation includes many more steps. The entire process is driven by profiles and by the format and

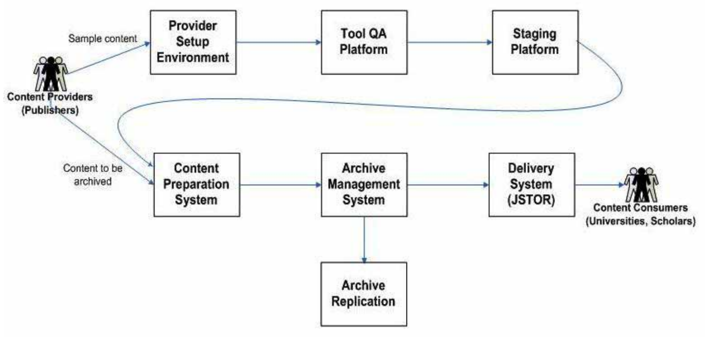
Sample content
Provider Setup Environment
Tool QA Platform
Staging Platform
Content Providers (Publishers)
Content to be archived
Content Preparation System
Archive Management System
Delivery System (JSTOR)
Content Consumers (Universities, Scholars)
Archive Replication

Figure 1 Portico System Overview

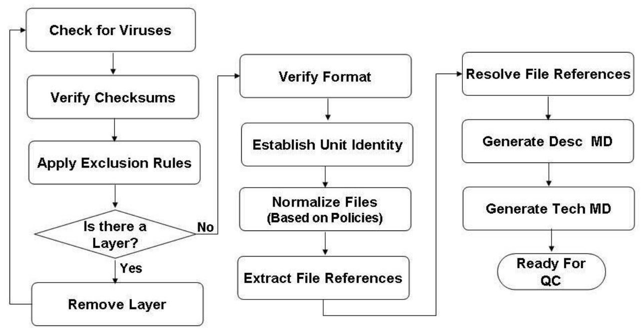

Figure 2 Automated Processing Steps

tools registries. A profile is a set of policies and rules that apply to a specific set of content. One profile might contain the file naming rules for Publisher ABC; another rules for Publisher XYZ. In some cases a single publisher has production processes and data streams that are so different that we maintain multiple profiles. The profile used for processing a given batch is captured as part of the events metadata.

Automated Processing Steps

The automated workflow is designed so that every file in a batch completes the first step before any file begins the second step. Although it is theoretically possible to do parallel processing, because of the dependencies between the article text and the referenced graphics and other files it is much easier to solve problems and keep the metadata in order if we proceed one step at a time. Therefore all the files in the batch must successfully complete one step before any can move on to the next step. That means that in some cases a batch will stop for human problem resolution and then continue processing.

Virus Check

The first step for all content is virus checking. We use ClamAV [12], an open-source virus scanning software. An important part of our event metadata is that the exact version of each tool used is captured and recorded; for ClamAV the event record includes the software version and virus signature file used as well as the date and time run.

Verify Checksums

Verifying checksums is straightforward. At present only one of our publisher participants currently provides checksums or any sort of reliable fixity check to verify the transmission and integrity of the data. We hope that the publishing community will move to adopt more robust processes in the future.

Apply Exclusion Rules

Exclusion rules specify that for a given content provider (publisher), certain classes of files should not be ingested into the archive. Ideally files that are not to be archived would not be sent to the archive at all. However some publishers have pre-existing content distribution systems and pre-defined content packages that we must accept for business reasons. Some examples of instances of exclusion would be when the file to be excluded represents an intermediate rather than final form or when the object is supplied in more than one format or resolution and only one format or resolution is to be archived. In one case a publisher has told us that none of their encapsulated PostScript files can be verified because they reference non-embedded, non-supplied fonts; they recommend that we preserve only the rasterized equivalents.

Remove Layer

"Layer" is the term used in PREMIS for both packaging, such as zip or tar, and encodings or encryption. Our automated workflow supports profile-driven expansion or removal of layers. The resulting new files have to go through the previous steps of virus check and checksum verification. In practice ClamAV and other virus checking software are capable of opening up some kinds of layers such as zip and tar and checking the contents. We ignore that capability and do virus checking separately for each file that resulted from the layer removal. This gives us more flexibility and cleaner metadata at the cost of some redundant processing. As our volume increases we will likely have to optimize this process.

Verify Format

Format verification combines identification and verification. A key decision in thinking about format is what to do when a file is damaged, not a fully valid instance of its purported format. We find it more useful to record that a file is, e.g., a "damaged" PDF than to record that it is a valid byte-stream. This makes format verification and identification a bit more complicated for us because JHOVE, our primary tool, can either verify whether a file is a valid instance of a given format or identify to what format a file does conform. That doesn't help with the "damaged PDF" problem or the wrong extension or MIME type problem, so we use JHOVE in combination with BSD file (a standard UNIX utility) in a multi-step process:

- Verify purported format based on MIME type using JHOVE
- If verification succeeds, record format and capture technical metadata
- If verification fails, attempt identification with BSD file
- If identified format is the same as purported format, the file is bad
- If identified format is not the same as the purported format, it might be mislabeled so verify the identified format again with JHOVE; if that fails again, the file is bad

This process is discussed in more detail in [10].

Establish Unit Identity

In the absence of any publishing industry standard for file naming and packaging for content distribution or exchange, each publisher has developed its own proprietary system. At this point in the workflow the original file names and paths are run through a regular-expression rules engine to identify which files go together to form an article. In the easiest case, content is organized in one directory per article:

```
\journalName\volumeX\issueY\article1
\journalName\volumeX\issueY\article2
…
```

Some publishers group files together by file format or function or combine hierarchical organization of the XML text files with communal directories for graphics. All of this is sorted out based on the rules in the profile for each publisher and the content reorganized into units by article or other unit of content (e.g., front matter or covers).

Normalize Files

Normalization of publisher-specific DTD instances to the NLM Archiving and Interchange DTD is done as early as possible in the automated processing so that subsequent steps which rely on the XML header or full-text file have only a single format to process. Although we do not currently do any graphics conversion, the same infrastructure would support converting other types of files during the automated processing.

Conversion of publisher DTDs to an industry-standard DTD is a complex process. Accordingly, the human portion of our process includes random sampling to detect problems that were not identified during the initial testing and programming. Also, a publisher DTD is likely to change over time driven by the publisher's business requirements. We therefore expect to have to update the conversion tools with some frequency. As with all tools used in the system, we track version numbers in the events metadata so that if a problem is discovered later it on, all the potentially affected content can be identified and corrected as necessary. The importance of this cannot be over emphasized. It has already proved invaluable to us in practice.

Extract / Resolve File References

Once the article texts have been normalized, we extract any references to external files (e.g., graphics) and try to locate the referenced files in the batch. This process is driven by another set of rules defined in the profile as each publisher may have a different way to do this and hidden business rules are common. In the best case filename references are complete and explicit; in some cases, however, considerable indirection may be involved. For example, the markup <dformula id="df27"> requires locating a file named "df27.gif"; <graphic filename="fig7"> may point to three graphics in three different directories: "fig7_thumb.gif" "fig7_mres.jpeg" and "fig7.eps". The task is to map from one reference to many and replace all the references with the correct archival accession IDs. We write the accession IDs directly into the normalized XML rather than maintain that information as a separate mapping table.

Generate Descriptive Metadata

One of the interesting characteristics of e-journal content is that descriptive metadata is abundant; in some cases there is too much metadata. E-journal articles supplied in marked-up SGML or XML (either full text or headers) normally have all the descriptive metadata clearly identified: author, title, journal, volume, issue, date, etc. In some cases publishers even include extra metadata not used directly in the article such as previous titles by which the journal was known or the identity of the copy editor or the date on which the proofs were mailed to the author. Some of this additional metadata is really the publisher's own business process data, not part of the published article. After consultation with the publisher we will remove that non-content information during conversion.

The descriptive metadata that goes into the Portico METS files is extracted from the NLM DTD article instances. It is then run through a light-weight automated curation process in which it is checked for required values such as ISSN and date of publication and ISSN and journal titles are validated against the master list of journals for which we have archiving agreements. This assures us that we are archiving only content for which we have a contract and also identifies cases where a title change has occurred if we have not already been notified of that by the publisher.

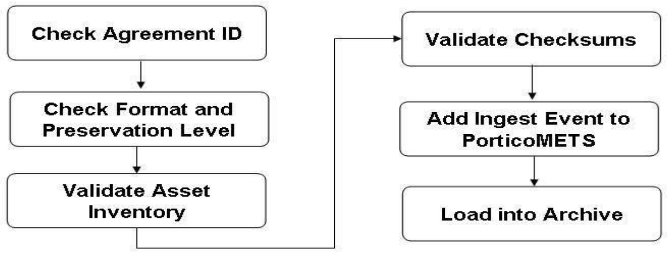

Figure 3 Archive Ingest

Release to Archive

The automated processing is followed by human inspection and problem resolution. After all problems have been successfully resolved, the content is packaged for release to the archive: new checksums are calculated, Portico METS files are created, one per article or equivalent, and are validated.

Archive Ingest

The content that has gone through the Content Preparation System "pre-processor" is loaded into the archive in the process shown in Figure 3.

Check Agreement ID

We have a business requirement that we only archive content for which we have appropriate archival rights, usually in the form of a signed contract with the rights holder or agent. Every file in the archive is traceable to an agreement; we even have a placeholder agreement to identify the rights in documents that we have created ourselves or which are in the public domain. In our current implementation, the contract or agreement constitutes the rights metadata for the archive. We expect that a more elaborate implementation of rights metadata tracking will be necessary in the future.

Check Format and Preservation Level

Every file in the archive is assigned a format name that points to an entry in the format registry. Every file is also assigned a preservation level: fully supported with promise of migration, supported with reasonable efforts only, or byte-preserved with no promise of migration. The preservation level is determined first from the format validity: a defective file cannot be fully supported; at best we can promise only reasonable efforts. Only a fully valid file can be fully supported.

During archival ingest we double check that the new content as described in the Portico METS files does not contain formats unknown to the archive's format registry and that it does not make preservation commitments that exceed the capabilities of the archive. This check is not strictly necessary right now as we control the "pre-processor" and the archive and keep them synchronized. We expect that in the future content may be pre-processed elsewhere or on other systems, in which case this check would be essential to ensure that we are not making commitments that we cannot fulfill.

Other Steps in Archival Ingest

The remaining steps in archival ingest are straight-forward: verify that all the files mentioned in the Portico METS file are present, verify that the checksums match, and finally update the Portico METS file to include the ingest date and time. Accession IDs were assigned in the Content Preparation System; in the future it may be necessary to move that to this point when content is pre-processed by other systems.

Future Directions

The Portico e-journal processing system was developed based on test data provided by the ten partner publishers who participated in the Portico pilot phase. Though the test data was chosen to cover a variety of different e-journal publishing systems and content delivery platforms, we expected that new problems would arise as we encounter new publishers. That has indeed proved to be the case. Among the enhancements now under development are

- Generating minimal descriptive metadata when no XML file is supplied (e.g., covers and front matter in PDF) by "borrowing" issue-level metadata from sibling articles
- Extracting issue level metadata from external manifest files
- Automatic deletion of extraneous unreferenced GIF files (too many GIFs)
- Shared directories of "library" GIF files referenced by many articles (too few GIFs)
- Automatically lowering the preservation level of well-formed but not valid PDF files to "Reasonable Effort"
- Automated repair for instances of formats with recognized and fixable problems
- Format validity of "Not Determined" so that we can archive content before all the necessary validation tools are available

We expect that this list will continue to grow for the next few years. As with any new system, use engenders feature enhancement requests.

Although the workflow described here was developed to process electronic journals, the underlying content model used to describe and manage the archived content is not journal-specific. All of the tools and many of the processing steps are sufficiently generalized that we expect to be able to support the processing of other content types on the same architecture. The details and sequence of the automated processing steps would likely change, however.

References

[1] Archiving Electronic Journals; Research Funded by the Andrew W. Mellon Foundation (Digital Library Federation and Council on Library and Information Resources, Washington, DC, 2003) http://www.diglib.org/preserve/ejp.htm

[2] E-Journal Archive DTD Feasability Study Prepared for the Harvard University Library Office for Information Systems E-Journal Archiving Project by Inera™ Incorporated (December 5, 2001) www.diglib.org/preserve/hadtdfs.pdf

[3] http://dtd.nlm.nih.gov/

[4] http://hul.harvard.edu/gdfr/

[5] http://www.loc.gov/standards/premis/

[6] http://www.schematron.com/

[7] www.cdlib.org/inside/diglib/ark/noid.pdf

[8] www.loc.gov/standards/mets

[9] http://hul.harvard.edu/jhove/

[10] Evan Owens et al., A Format-registry Based Automated Workflow for the Ingest and Preservation of Electronic Journals, Digital Library Federation Fall Forum 2005 http://www.diglib.org/forums/fall2005/ or http://www.portico.org/about/Portico%20DLF%20Fall%202005.pdf

[11] http://www.jstor.org/

[12] http://www.clamav.net/

Author Biography

Evan Owens is Chief Technology Officer of Portico. Prior to joining Portico in 2003 he worked for the University of Chicago Press for over twenty years in various roles including IT Manager and Electronic Publishing Manager of the Journals Division. He has been a frequent speaker at conferences in the US and in Europe on publishing technology topics including SGML/XML for journal publishing, SGML/XML editing and typesetting workflows, and peer-review systems