Fedora and the
Preservation of University Records Project

2.3 Ingest Tools

Version
1.0

Date
September 2006

Digital Collections and Archives, Tufts University
Manuscripts & Archives, Yale University

An Electronic Record Research Grant funded by the
National Historical Publications and Records Commission

Digital Collections and Archives
Tisch Library Building
Tufts University
Medford, Massachusetts 02155
http://dca.tufts.edu

Manuscripts and Archives
Yale University Library
Yale University
P.O. Box 208240
New Haven, Connecticut 06520-8240
http://www.library.yale.edu/mssa/

Co-Principle Investigators
Kevin Glick, Yale University
Eliot Wilczek, Tufts University

Project Analyst
Robert Dockins, Tufts University

This document is available online at
http://dl.tufts.edu/view_pdf.jsp?urn=tufts:central:dca:UA069:UA069.004.001.00008
(September 2006)

Fedora and the Preservation of University Records Project Website at
http://dca.tufts.edu/features/nhprc/index.html

Funded by the
National Historical Publications and Records Commission
Grant Number 2004-083

Fedora and the Preservation of University Records Project

PART ONE: INTRODUCTION

PART TWO: INGEST

PART THREE: MAINTAIN

PART FOUR: FINDINGS

TABLE OF CONTENTS

OVERVIEW

Although it was not the original focus of research, the project team utilized the expertise of its members to undertake some development work into some of the specific tools needed to support the accessioning of electronic records into a preservation repository. Such tools are necessary to make Fedora more suitable for preservation of university electronic records and more compliant with OAIS model specifications. The work was undertaken separately at both Tufts and Yale under the supervision of Robert Dockins, the Project Analyst. Two particular tools are described below in detail.

TUFTS INGEST PROTOTYPE SYSTEM

In order to support the work described in “Ingest Projects,” the Tufts project team developed a prototype ingest system which automates many steps of the ingest process. The system, known as the Tufts Ingest Prototype System (TIPS), allowed the project team to gain experience with the processes involved in production-scale ingests. The project team developed TIPS concurrently with the Ingest Guide, allowing theoretical knowledge and practical experience to inform the development of the Guide and the tool.

TIPS is available for use under the Mozilla Public License Version 1.1. It can be downloaded from http://dca.tufts.edu/features/nhprc/reports/tips/index.html. Archives can use TIPS to execute many of the tasks described in Section B Transfer and Validation of the Ingest Guide. However, this is not a production-ready tool with a polished user interface. It is a developer-oriented tool that can help archives develop scalable transfer and validation workflows within an ingest system. Although the project team used TIPS with a Fedora-based repository, it is not a Fedora-specific tool.

TIPS defines a Submission Information Package (SIP) format based on the popular info-zip compression format with an XML manifest. TIPS uses the manifest to group together related files into “digital objects,” provides fixity information in the form of checksums, and allows the SIP creator to digitally sign the manifest. TIPS includes an API for creating and validating SIPs in this format. This SIP format is well-defined and an XML schema exists for the manifest.

The project staff created an XML format for submission agreements, although it is less well defined than the XML format for the SIP and there is no formal schema. Submission agreements are centered on the idea of “submission elements.” Submission elements are sub-parts of a submission agreement representing a set of items in an accession that share certain properties, such as format types, record types, and access restrictions. Producers, sometimes with the help of archives, assign objects to particular submission elements during SIP creation. The decision about assignment to a submission element is largely an intellectual question. In the absence of rich metadata, it will probably need to be done manually; however, if a recordkeeping system has sufficient metadata, submission element selection may be automated.

The object components assigned to a particular submission element do not all have to have the same file format (however, they must all belong to a pre-determined set of file formats). To handle different format types, a submission element is associated with one or more “object profiles.” Object profiles provide information about a format type, including the ability to recognize a file of that type and specific validation and transformation procedures for files of that format.

After a SIP is accepted by TIPS, each object is assigned to its submission element. Then, it is tested against each object profile in turn. The object is bound to the first profile it matches, which then determines the validation and transformation that object will undergo.

After a digital object is transformed and validated, it must be submitted to the preservation system. TIPS calls preservation systems “repositories” and defines an API for arbitrary repositories. We implemented this API for both Fedora 1.2.1 and Fedora 2.0 by binding to the API-M and API-A soap interfaces of Fedora.

Thus by writing object profiles, and combining them together into a submission agreement, the project team was able to define all the actions that should occur to an object before being submitted to a preservation repository.

TIPS does have a number of quirks. Most importantly, the project team constantly changed the design of the tool as the team built it concurrently with the development of the Ingest Guide, which itself underwent many revisions. There are a number of places were ideas were tried out and abandoned, but affected the way the project team shaped the code. Furthermore, object profiles are simply implemented as dynamic scripts that are run on demand. The scripting system is quite fragile and difficult to work with. The task of writing object profiles is complicated by the lack of a central repository of file format information with globally unique identifiers. Such a system would make it possible to register handlers that recognize, transform, and validate files of given, well-specified types; as it is the project team had to make do with what it had. Finally, TIPS is driven by an embedded workflow engine, with the idea that the operation of TIPS could be customized by writing custom workflows. However, the workflow integration greatly complicated TIPS for little additional benefit, and it interacts badly with the transaction management subsystem, leading to difficult-to-find bugs.

TIPS is therefore unwieldy and unsuited for large-scale use. It has been, however, a great help in discovering the issues that one will confront when attempting to build a highly customizable, highly automated system for ingesting electronic records in a trustworthy manner.

YALE EUDORA EMAIL INGEST TOOL

The Yale project team investigated the issues surrounding ingest of university records in the form of email, a problem of both file format and workflow common to a number of universities. In particular, the project team sought a solution to the scalability issues created by email stored in thousands of staff workstations using a proprietary software application that stores the email in a proprietary format.

Yale has a robust email operation, delivering over 500,000 messages everyday, excluding spam. The University has a diverse email environment with numerous email servers and many email software packages in widespread use. There is no single mandatory supported interface or application for utilizing the email services. AppleMail, Thunderbird, Eudora, Pine, Webmail, and the Outlook family of products are all widely used by faculty, staff and students on campus. These programs access the central email servers and store the resulting email messages in different manners that pose different issues for recordkeeping and preservation.

The most common method for connecting to email services from Yale staff workstations is using email application clients and the Post Office Protocol version 3 (POP3), an application-layer Internet standard protocol, to retrieve email from a remote server over a TCP/IP connection. POP3 allows users to retrieve email when connected and then to view and manipulate the retrieved messages without needing to stay connected. Although most client applications have an option to leave mail on server, Yale users employing POP3 clients generally connect, retrieve all messages, store them on their workstation as new messages, delete them from the server, and then disconnect. Far fewer Yale staff utilize, the Internet Message Access Protocol (IMAP) with their email applications. IMAP supports both *connected* and *disconnected* modes of operation. Those staff that do utilize IMAP generally leave messages on the server until the user explicitly deletes them. This is a reason that IMAP is more common in situations where multiple staff members share a mailbox. The fundamental difference between POP3 and IMAP when university records are concerned is that POP3 offers access to a mail drop; the mail exists on the server until it is collected by staff member’s email client application. Even if the staff member leaves some or all messages on the server, the staff member’s messages store is considered authoritative. In contrast, IMAP offers access to the central mail store; the staff member’s client application may store local copies of the messages, but these are considered to be a temporary cache; the central server’s store is authoritative.[1]

The project team was most interested in the issues posed by Yale staff using the Eudora email client application in a POP3 configuration. Other projects have focused on managing email at the central server[2] or with the Outlook family of applications.[3] Both of these solutions were

[1] Wikipedia contributors, “Post Office Protocol,” *Wikipedia, The Free Encyclopedia*, <http://en.wikipedia.org/w/index.php?title=Post_Office_Protocol&oldid=77082274>.

[2] Marlan Green, Sue Soy, Stan Gunn, and Patricia Galloway, “Coming to TERM: Designing the Texas Email Repository Model,” *D-Lib Magazine*, Volume 8, Number 9 (September 2002), <http://www.dlib.org/dlib/september02/galloway/09galloway.html>.

[3] Maureen Potter, “XML For Digital Preservation: XML Implementation Options for E-Mails,” 2002 <http://www.digitaleduurzaamheid.nl/bibliotheek/docs/email-xml-imp.pdf#search=%22dutch%20testbed%20outlook%20email%22>.

insufficient for the current Yale environment because they either supposed a level of control over central recordkeeping that does not exist at Yale or require a level of recordkeeping intervention by record's creator that is not practical. It was determined that the most prominent university officers were using the Eudora email client application. The recordkeeping problem is that Eudora stores email not as individual files, but instead bound together with other files from the same mailbox. Every mailbox created in Eudora is stored as two different proprietary files that work together to give access to the individual emails. Because it is at this client application level that the email is set aside, organized, and stored as records, it is necessary to accession the email from the client application. This poses great sustainability problems. First, the file format of the mailbox files is proprietary and in no way would be considered a de jure or de facto standard. The decision was made that the file format would need to be converted to one more stable. The second issue is that the email is distributed across hundreds of computers on campus, requiring significant resources to maintain and/or transfer.

The Yale project team designed software to address the problems for Ingest created by the Yale use of Eudora. The concept of the operation is that the Archive enters into an agreement with Producer. They give the Archive access to their Eudora mailboxes, either on their workstation, or on a network folder. The Archive agrees to copy all messages from an agreed upon set of email folders (e.g. not inbox, outbox, sent, or trash). The Producer organizes their email folders and discards inconsequential messages (this operation would only apply to email that an employee sends or receives as part of his/her work at Yale). The Archive would periodically gain access to each Producer's Eudora mailboxes. Each mailbox file would be copied. Once copied, each mailbox file would be parsed to separate the out the mail messages. The separate mail messages would be saved as plain text and marked up in XML. Header information and other documentation would be copied into metadata. Attachments from the "Attachments" directory, are copied and linked to corresponding email message. All records would be transferred to either a compliant recordkeeping system or preservation system. The folder structure of the Producer's mailboxes is replicated in the recordkeeping and preservation system. When the operation is undertaken subsequently, messages already copied are ignored, and new messages are transferred. The tools were built as a series of scripts, in both Perl and Python, so that it could be easily configured and scheduled to run in an automated fashion. The path into Fedora utilized the Vital Batch Import module created by VTLS.[4] The toolset was designed in a modular fashion, relying on open source components, so that different features could be more easily swapped in and out during development. For example, the tool was utilized to output records to both the Yale's electronic recordkeeping application (Livelink), as well as the preservation application (Fedora).

While the development of this email ingest tool was helpful in understanding some of the issues of ingesting into Fedora, there are still a number of issues with the toolset in its current state. There is no user interface to the software. In order to configure or run the scripts a user must be comfortable enough working in the code itself. The scripts require access to the email folders of university staff members, often high-level staff, a level of access that the Yale University Archives is not normally granted. Also, the scripts must be either installed onto Producer workstations (which resulted in user permissions and scalability issues) or run from a central

[4] For more information on Fedora integration products offered by VTLS, see <http://www.vtls.com/Products/vital.shtml>.

administrative server (which resulted in processing and network traffic performance issues). These problems may be quite simple to resolve, but such work fell outside the scope of this project.