Transfer and Inventory Components of Developing Repository Services

Leslie Johnston, Library of Congress, Office of Strategic Initiatives
lesliej@loc.gov

In recent work at the Library of Congress, we have determined that our most basic Repository needs are really basic: How do we know what we have, where it is, and who it belongs to? And how do we get files – new and legacy – from where they are to where they need to be? The Library of Congress Office of Strategic Initiatives has been working on solutions for a category of activities that we refer to as “Transfer.” At a high level, we define transfer as including the following human- and machine-performed tasks:

- Adding digital content to the collections, whether from an external partner or created at LC;
- Moving digital content between storage systems (external and internal);
- Review of digital files for fixity, quality and/or authoritativeness; and
- Inventorying and recording transfer life cycle events for digital files.

The work on transfer has focused primarily on work with external partners, including those that are part of National Digital Information Infrastructure and Preservation Program (NDIIPP) (Anderson, 2008); the National Digital Newspaper Program (NDNP) (Littman, 2007); the World Digital Library (WDL); and the Library’s Web Capture initiatives. Content transfers to the Library have largely consisted of small numbers of bulk transfers of content – from tens of gigabytes to over one terabyte at a time – using initially manual processes driven by Library staff to pull packages of content into the Library’s environment.

A package is defined as a set of files stored in a file system, which may be a subset of a larger collection of content, to be transferred and managed as a unit. The set of files comprising a package may be transferred as a single file in a container format such as ZIP or tar to be unpacked upon receipt. Working with John Kunze of the California Digital Library, Andy Boyko, Justin Littman, Liz Madden, and Brian Vargas of the Library produced a generalized version of what had been initially referred to as the "LC Package Specification,” now called "**BagIt**."

The base directory of a Bag contains a **file manifest**, a **content directory**, and an optional **package information** directory. The **content directory** contains the contents of the package, as defined by its producer. The content directory may have any name and internal structure. There is no limit on the number of files or directories this directory may contain, but its size should make practical transfers easier, based on physical media limitations or expected network transfer rates. The file manifest lists the names and checksums of the content files and the package information files, excluding itself and any shipping files. Any commonly recognized checksum algorithm can be used to generate the manifest. Neither the file manifest nor the package manifest obviates the need for descriptive metadata being supplied by the package producer. The manifests assist in the transfer and archiving of the package as a unit, rather than supplying any description of the content.

The Library has created a **Bag Validator script**, which checks that a Bag meets that standards of the specification; that all files listed in manifest are in the data directory; that there are no files in the data directory that are not listed in manifest; and that there are no duplicate entries in the manifest. The **VerifyIt script** is used to verify the checksums of files in a Bag against its manifest every time the files are moved or copied. The Bag Validator is a Python script and VerifyIt is a shell script.

A client-side **Bagger** application has been developed to assist partners engaged in small-scale deposit transfers, automating the packaging and submission of small amounts of content without requiring Library involvement, and ideally requiring no client-side IT support or infrastructure. It is implemented as a Java Web-Start application for use across platforms, and supports the aggregation of files into Bag packages, including the creation of checksum manifests and Bag information files. This application was in part built on top of **BIL**—the BagIt Library—a Java library developed to support Bag services.

In order to support the expanding numbers and types of transfers, several software tools were needed to help automate transfers. The ***Deposit service*** is a web-hosted application for use by transfer partners in registering a new transfer; this application will support the registration and initiation of the transfer content via network transfer (rsync, ftp, https), and via fixed media, such as hard drives or DVDs. The web application is implemented using the Django web framework. At the time of this writing, Deposit services are mid-way through the production implementation process, including review by representatives of the multiple digital content acquisition projects.

The Deposit tools are tied to a series of ***Transfer*** and ***Transport*** back-end tools used for retrieval, receiving and managing of content transfers. The **Parallel Retriever** implements a simple Python-based wrapper around wget and rsync, capturing files and producing a package that meets the BagIt specification when given a "file manifest" and a "fetch.txt" file. It was initially built specifically for rsync, but has been extended to HTTP and FTP.

Underlying “Core Transfer” components support various transfer functions. The components are completely independent of any workflow, though they may of course be invoked by any designated workflow. The ***Inventory System*** has three parts: the Package Modeler, a suite of command line inventory tools, and a reporting web application. The Package Modeler is implemented using Java objects mapped to a PostgreSQL database using Hibernate for object-relational mapping. The Inventory tools inspect packages and update the Package Modeler, calling the command line tools. The reporting web application allows users to view reports on packages. Since the Package Modeler must also represent the history of a package, it records events that occur on a Package level and on a file level.

The goal in developing Inventory Tools is to satisfy needs identified through the process of doing transfers manually and attempting to record their outcomes. These include keeping track of package transfers for a project, tracking individual packages and events associated with them, and a list of the files that make up each package and their locations. For legacy collections these tools can be pointed at existing directories to package, checksum, and record inventory events to bring the files under initial control.

The Transfer, Transport, and Inventory tools can be tied together into any of a number of project specific **Workflow systems**. The underlying workflow engine is jBPM, an open-source workflow system. The drivers of a workflow are process definitions, which represent the process steps. jBPM Process Definition Language (jPDL), the native process definition language of jBPM, is used to encode the workflow process steps as XML. A workflow can be designed using the visual editor Graphical Process Designer, a plug-in for the Eclipse platform. The first web user interface, built for NDNP, allows users to identify lists of tasks to be performed, initiate, monitor and administer processes; and notify the workflow engine of the outcome of manual tasks, including task completion. Workflow tasks instantiated through the system include transfer, validation by an NDNP-specific validation application (Littman 2006), manual quality review inspection, and file copying to archival storage and production storage. The Transfer UI was implemented using Spring MVC.

At the time of this writing, the Transfer, Transport, Inventory and Workflow services have been put into production for NDNP (Littman, 2009). The Transfer and Transport tools have been put into production for NDIIPP, and production implementation is under way for the Inventory, Deposit, and Workflow services, as well as the Bagger application. By summer 2009 many incoming collections will be processed using these tools, and a retrospective inventory of the Library's existing digital collections will be undertaken in 2009 using these tools.

The first of the Transfer tools—the Parallel Retriever, the Bag Validator, and VerifyIt—have also been released by the Library of Congress as open source on SourceForge (http://sourceforge.net/projects/loc-xferutils/). Additional tools and utilities will be released over time.

Why are such transfer tools and processes so important? Transfer processes are not surprisingly linked with preservation, as the tasks performed during the transfer of files must follow a documented workflow and be recorded in order to mitigate preservation risks. Defining, implementing, and documenting appropriate transfer processes depends on the requirements of each collection building project, which can vary wildly. While our initial interest in this problem space came from the need to better manage transfers from external partners to the Library, the transfer and transport of files within the organization for the purpose of archiving, transformation, and delivery is an increasingly large part of daily operations. The digitization of an item can create one or hundreds of files, each of which might have many derivative versions, and which might reside in multiple locations simultaneously to serve different purposes. Developing tools to manage such transfer tasks reduce the number of tasks performed and tracked by humans, and automatically provides for the validation and verification of files with each transfer event.

Why are we looking at close integration between transfer and inventory functions? Inventorying and audit functions have been identified as a vital aspect of data curation. Inventory services can bring several benefits, including collection risk assessment and storage infrastructure audits. Realizing any benefits for effective data management relies on knowledge of data holdings. Knowledge of file-level holdings and recording of life cycle events related to those files from the moment that they enter the collection and in every future action reduces future risk by storing information that can be used in discovery, assessment, and recovery if and when a failure occurs.

Identifying needed services as modular rather than monolithic has allowed the Library of Congress to research and implement each of these functions in a more nimble way, all the while planning to fit those services into a larger scheme of repository services. The integration of modular transfer and inventory services as well as workflows allows for separation of tasks based on project or collection or format needs while supporting backend data integration where required. Modules can be idependently reimplemented in the future when the need arises. This also allows for extensions to services and functionality that we have not yet even considered, let alone planned for.

These modular services do yet not equate to everything needed to call a system a repository. There are no detached end-user discovery and delivery applications. Descriptive metadata is not yet tracked with the media files. There are currently no granular rights and access policies nor means to enforce them. Preservation monitoring is not yet in place. But there is a set of services that equate to many aspects of "ingest" and "archiving" – the registry of a deposit activity, the controlled transfer and transport of files, and an inventory system that can be used to track files, record events in those files' life cycles, and provide basic file-level discovery and auditing. Through the Inventory tools we expect to be able to provide persistent access at a file level. In other words, it may not yet be a full-blown repository, but is the first stage in the development of a suite of tools to help the Library ensure long-term stewardship of its digital assets.

References

Anderson, Martha. 2008. Evolving a Network of Networks: The Experience of Partnerships in the National Digital Information Infrastructure and Preservation Program. The International Journal of Digital Curation (July 2008: Volume 3, Issue 1). http://www.ijdc.net/ijdc/article/view/59/60.

Littman, Justin. 2006. A Technical Approach and Distributed Model for Validation of Digital Objects. D-Lib Magazine (May 2006: Volume 12, Number 5). http://www.dlib.org/dlib/may06/littman/05littman.html.

Littman, Justin. 2007. Actualized Preservation Threats: Practical Lessons from Chronicling America. D-Lib Magazine (July/August 2007: Volume 13, Number 7/8). http://www.dlib.org/dlib/july07/littman/07littman.html.

Littman, Justin. 2009. A Set of Transfer-Related Services. D-Lib Magazine (January/February 2009: Volume 15, Number 1/2). http://dlib.org/dlib/january09/littman/01littman.html.