HistoryDiscussion

Discussion of a History System for DSpace

Those who cannot learn from history are doomed to repeat it.
– attrib. George Santayana

Please use this page to share ideas about the History System and how you want it to evolve. Think of it as an initial design specification (along with background material) for a new history mechanism.

Topics

- #toc01 What is the History System, and History?
- Usage Scenarios
- #toc03 Design Considerations
- #toc04 Relevant Issues in More Depth
- #toc05 Next Steps

What is the History System, and History?

The old DSpace History System (up to Release 1.4) was intended to create a log of all modifications to the archive’s object model, encoded in RDF. The record is stored in XML in many text files, broken up approximately by transaction.

It was intended to be a source of provenance metadata for the objects, but as far as we know it has never actually been used. Many sites disable history, to save disk space and increase reliability.

Provenance metadata documents the origin, and subsequent handling, of archival assets such as Items and Bitstreams. Provenance is sometimes considered a synonym for the history of an object.

DSpace needs provenance metadata more than ever. There is a growing interest in digital preservation, and there are even some preservation experiments based on DSpace. To that end, we are revisiting the design of the history mechanism from the ground up, so we can consider all the research, standards, and practical experience that have become available since the original design was done.

The History of History

The History System has been essentially unchanged from the first release through 1.4. It is implemented entirely in the class `org.dspace.history.HistoryManager`. Every block of code in the DSpace implementation that makes a change to the data model, also invokes the static `HistoryManager.saveHistory` method to add that "event" to the history record.

It creates an RDF model of the event, expressed in an early version of the ABC Harmony data model and ontology.

For a more thorough description of the existing system, see this page of the DSpace system documentation.

Unfortunately, flaws in the design and implementation make most of the current history data unusable for provenance:
The RDF encoding has several syntactic and namespace problems. 
- The records of an object's state before and after each event cannot be connected to the records of events.
- All DSpace objects are referenced by local database identifiers instead of persistent identifiers.

The Simile Project paper 
[DSpace History System: Descriptive Note][http://simile.mit.edu/reports/dspace-history/description.pdf] covers the specific problems with the present system more thoroughly, and compares it to the RDF conventions and best practices of the time.

There is some salvageable information in the old history records. It should be possible to extract a coarse-grained sequence of events for each DSpace object, and convert it into a new format.

**Why Record History: Usage Scenarios**

The first thing to consider when proposing a replacement for the History System is, 
*"Why do we want this data and what are we going to want to do with it?"*

In general, the answer is "History is a source of provenance metadata for objects in the archive."

Provenance is a form of *preservation metadata*, which is supposed to assist in maintaining "the viability, renderability, and understandability of digital resources over the long-term".

Unfortunately, we have a dearth of proven implementations or examples of successful digital preservation from which to learn, so there are no guidelines to follow, informed by actual experience.

Although the design of a history system *should* be driven by detailed use cases, we don't have any relevant examples available, nor the resources to develop our own. In lieu of true use cases, here are some scenarios, both hypothetical and based on similar experiences:

1. Recover metadata after erroneous change

In this scenario, an administrator mistakenly modifies the descriptive metadata (i.e. DC) on many items, replacing a certain field with the wrong value (so the old value is lost).

After months or years, the error is discovered. There is no pattern to the damage that would help us search out the affected items, so search tools are no help. It's also too late to consult web server logs (which may not have had enough detail anyway).

However, the information we need to find the affected items is in the History. Searching History records for all events where the damaged metadata field was changed by the administrator who committed the errors should yield all of the affected items with few false positives.

We can even undo the damage if the History records include the state of metadata fields before and after the change. This is an example of where saving more information than seems to be immediately useful can have unexpected benefits to later "preservation" efforts.

Note how History is used as a lightweight "versioning" system here. This only works if it records the state of an object before it is changed, and this may only be practical for data that aren't too bulky, like values of metadata fields. Saving the state of bitstreams that get altered or deleted may not be possible because of the volume of storage that would demand.

2. Detect alterations to the content of a thesis
An electronically-submitted thesis is (someday) the document of record. How can you prove that its contents have not been modified since the original submission? Does the History record identify the submitter in a secure and non-repudiatable way? Can it supply a chain of custody for the document? Of the formats available, which is the original, and how were the others derived (are they complete and accurate equivalents)?

To answer this question properly, the history record has to be "provably" accurate and complete. We would need to have some confidence that no events are missing, and that the record itself hasn't been tampered with.

The format provenance question is fairly simple to answer: For each Bitstream in the Item, show the chain of events affecting it. It will be obvious if it was part of the original submission, and if it has been altered or replaced since then.

3. Intrusion detection

Can History help monitor a DSpace archive for possibly unauthorized or questionable activity?

Since the history system is already recording all events that affect the archive, ideally in an unalterable and secure manner, it is the ideal spot to monitor. Instead of monitoring each client of the DSpace API separately, or adding another layer of instrumentation to the Business Logic layer, we can watch a stream of events that is already available.

Interpreting the meaning of History events should also be a lot simpler than e.g. reverse-engineering the traces left in Web server logs to detect changes to the archive. It might be helpful to correlate History events with other logs (e.g. through accurate and precise timestamps) to obtain other application-specific details such as client IP addresses.

A typical analysis task would be to plot the activity of each EPerson by time-of-day and day-of-week, and look for events in exceptional (or abnormal) time periods.

The only drawback is that History probably will not track "read" events such as disseminations, since it does not need them for provenance. Read events can be very interesting for intrusion detection. The security analyst will have to get read events from some other source if History does not record them.

4. Detect format migrations

You want to know where each of an Item's Bitstreams came from. Which ones were part of the original submission (or manually added later by an administrator), and which were generated automatically. Of the latter, are they "derived" Bitstreams created by e.g. a 'MediaFilter' plugin for a special purpose, like thumbnails, or true "transformations" of an original digital object to an equivalent object in a different format.

This question is similar to #2, without the veil of suspicion of fraud and with attention to a more general answer.

Why consult history data instead of the Bitstream's technical MD? The DSpace architecture currently has no rigorous way to show how or why a Bitstream was created. Each application, such as an Ingester and a Media "Filter, has its own way of indicating the provenance of the Bitstreams it creates, sometimes in subtle ways. If an Item is transferred to another DSpace archive, the minor losses incurred in exporting and re-ingesting it might erase these clues. So long as the Item's History data is preserved through that transfer, however, it will remain an accurate source of provenance.

Another advantage: The History system is integrated at a low level in the object model so it captures every change to the Item in the same way. Instead of relying on applications to leave clues about the provenance of Bitstreams they create, we have one uniform source of data.
5. Locating Items Ingested From a Faulty Submission Application

One of the applications that submits new content to the archive has a serious bug, but it isn't discovered for some months. After it has been diagnosed and fixed, the archive administrator has to identify the items created by this application so they can be repaired or re-submitted.

Since the submission interface was designed to deposit an accurate and unembellished record of the content it is given, nothing in the state of the Items it creates betrays their origin.

Fortunately, the history record of events for the creation of those items mentions the application responsible. A search through history records for this application will identify all the affected items.

6. Tracking Availability of a Document for Legal Requirements

Quoting from a scenario TimDonohue said “was mentioned by UIUC legal counsel”:

A faculty member has loaded a technical report into DSpace but has chosen (or is required) to restrict access to the local campus for six months. After the six months is up, the technical report is made available worldwide. A patent attorney is interested in knowing exactly when that technical report was made available worldwide because it shows evidence of prior work that would effect his client's (not the faculty member) ability to get a patent.

This requires the History System to track the state of access controls on an Item, so we can tell its exposure at any point in time. Given that, the attorney could get her answer from a report of all the events that changed access controls on that Item.

Design Considerations

This is an enumeration of topics to consider in the design of a new history system, and the questions the design should answer. I've presented it in the form of an outline to organize the categories of design considerations more clearly. Some issues that need to be pursued in more depth are presented after the outline.

- **Scope of History Data** - choosing which events and details are recorded.
- Guiding principles:
  - History is a record of the events and actions that modified the contents and metadata of the archive. It does not necessarily include the actual changes.
  - Since we have little experience or existing practice for guidance, err on the side of “saving too much” in the hope of being more helpful to future consumers.
  - Temper this principle with the admonition against saving anything too bulky to be practical (e.g. old versions of content), or obviously irrelevant to provenance.
  - Ask first if data is relevant to the mission of the history system, rather than if it is inherently “future-proof”.
  - Even obselete data (e.g. long-gone IP addresses) can be significant in context.
  - We cannot predict what future preservationists will find useful.
  - “History is only about what we know is true.” That means only events occurring in DSpace (which may include other federated DSpace archives).
  - Bear in mind that the profile of desired history data will change over time, as DSpace itself and the nature of its contents evolves.
  - History is not a versioning system. They are distinct, separate problems.

- Which Events in an object's life cycle are relevant to provenance?
• Origin: "born digital", or created from (or about) a physical object?
• History does not cover the time before the Item and Bitstreams were submitted to the archive?
• Remember, History is only about what happens in the DSpace archive.
• Not important for "born digital" objects.
• Digitized physical object (e.g. scanned image) should include metadata about the digitization process, but store it as bitstream-specific technical/provenance metadata.
• DO NOT "import" or crosswalk history/provenance data from another archive (or another DSpace), but store it with object as "foreign" metadata.
• Ingest or Creation
• Was object submitted by the creator as an original work? (Content only)
• .or, Custody transferred from another archive or repository/CMS?
• .or, Replica (for backup) of a master copy stored elsewhere?
• Object is Resident in the archive
• Alterations to content, object hierarchy, or metadata.
• Only track changes to some metadata fields?
• Changes to access control: e.g. visible to everyone, or restricted?
• Preservation events: Format derivations and format migrations.
• Re-binding object to a new (or additional) persistent identifier.
• Do NOT record: Dissemination events (even migrating copies in other repositories).
• Leaving the archive
• Transfer of custody to another archive.
• .or, Discard this backup copy of an object whose master copy is archived elsewhere.
• .or, Destroy the only archived copy of an object.
• Replication or transfer of objects between DSpace archives
• Only if it is triggered by a policy, for the purpose of preservation.
• Helpful to have a record at both source and target sites, so former home knows what happened to a transferred object.

• Subject scope: what kind of objects?
  • Bitstream
  • Item
  • Collection
  • Community
  • (Maybe) Transient objects like Workspace`Item, Workflow`Item - if we want to record workflow approvers.

• Data scope: What details of an Event to record.
  • "Outcome" of each event, identify which objects were altered.
  • Actual changes to descriptive metadata in the object model.
  • Actual changes to technical metadata in the object model.
  • (MAYBE) Actual changes to relationships between Bitstreams in an Item - this is administrative metadata?
  • Indicate changes to rights metadata (license) – maybe actual content change.
  • Note: Depends on place of rights metadata in content model, how well it is integrated.

• Granularity: How much activity does one event cover:
  • Atomic change to one object in the DSpace object model.
  • (MAYBE) Define boundaries of a "transaction" with respect to client application, tag all events that belong to that transaction.

• What is outside the scope of History?
  • DO NOT record attempted transactions that fail and get rolled back.
  • (MAYBE) DO NOT record events on "transient" objects such as `WorkspaceItem`, `WorkflowItem` – except perhaps workflow approvals?
  • DO NOT record some changes to descriptive metadata; perhaps this has to be configurable.

• Content (Schema) of History Data - exactly what about an event is recorded?
  • One type of datum, the "event", with contents:
    • DSpace archive instance where event occurred. (Needs global identifier)
    • Object that was acted on, "subject" of the event.
    • Type of the action (creation, modification, etc).
    • Date and Time it occurred.
    • (MAYBE) Responsible EPerson (or agent) - if possible! (global persistent name may not be available)
    • Outcome - details of changes made.
    • Additional DSpace objects acted upon, e.g. Collection that Item was added to.
    • Name-and-versions of software modules involved (both DSpace-internal and "application", if available).
    • (Optional) Identify the higher-level transaction this event belongs to, to group concurrent events.
    • (MAYBE) Digital signature to mark the authenticity of the event record.

• Security Considerations
  • Authenticity - prove the history was recorded by a particular DSpace archive.
  • Completeness - "prove" no events are missing - difficult!
  • Fixity - Check for errors in data storage and recall.
  • Access control: Initially, only DSpace Administrator group is allowed to read History.
• Performance Considerations

  • Recording should be fast: although submissions and changes are likely to be a small fraction of the activity of an archive, they often come in large batches (mass ingests or format migrations). Adding even seconds to a transaction would cause a noticeable delay when processing a batch of thousands of them.
  • Space is also a concern, for the same reason. History data is intended to be part of the permanent record so it should not grow unreasonably large.
  • Recording history must be efficient so there is little burden on the DSpace server. Some site administrators may not see the benefit in having history (its value is still a matter of conjecture, after all), so there should not be any compelling reasons to eliminate it.
  • History record does not need to be available in real time, but any lag should be a matter of hours, not days.

• Implementation choices

  • Internal representation:
    • Should be driven by how data is most often accessed:
      • Most frequent: Answering global queries.
      • Rare: Local to objects - but maybe less rare when writing out AIPs.
    • What best insures the preservation of the history data itself?
      • i.e. flat files vs. database; use of open standards.
    • Reified storage, both online triple-store and written out in AIPs.
    • Format: Compare RDF (based on ABC Harmony?) with PREMIS Event model.
    • Representation stored internally need not be the same as the representation in an AIP, SIP, or DIP.
    • Internal format should be extensible and capable of more detail and rigor than external formats, which favors e.g. RDF.
    • Consider storing two separate representations (e.g. flat files and RDBMS):
      • Reconcile them periodically as a preservation test.
    • RDBMS allows efficient data access and queries.
    • Need tools to check and validate history data
    • Check format, like archive’s ChecksumTool
    • Sanity-check for consistency and completeness, hard problem.
    • Configurable scope, options to control which events and what level of detail gets recorded.
    • Integration with present and future DSpace data model.
    • How will history fit into AIP model?
      • Consider adding an "event" system with pluggable event monitors to implement event recording (compare with Robert Tansley’s publish/subscribe recommendation).

• Exposing (Disseminating) history data

  • APIs and protocols, e.g. Sesame SOAP interface to connect to Longwell2 for RDF presentation and browsing.
  • What sort of queries and dissemination requests have to be answered? (e.g. by time, by object, by EPerson, etc.)
  • Desired import and export data formats for history? History in SIP and DIP (e.g. METS `digiprovMD` element).
  • How is History data correlated with other data and metadata?
  • Matching persistent identifiers of DSpace objects, EPerso, etc. in a way that is meaningful and preservation-oriented.
  • Correlating timestamps with other event logs – need generalized event mechanism that includes consistent time.

• Relevant standards and publications

  • OAIS reference model, specifically PDI
  • PREMIS event model
  • RDF, and ABC Harmony ontology
  • METS home page
  • RLG preservation activities
  • CIDOC CRM, for an example of a preservation data framework.

Anchor(toc04)

Relevant Issues in More Depth

This section lists some relevant issues that are not completely resolved in the current state (and perhaps future development direction) of DSpace.

Perhaps these entries can inform other discussions, or people who understand the issues better than I do will fill in some answers here.

• Ownership – How can we tell who owns, or is responsible for an object?
  • Owner or custodian of object is important to provenance and history.
  • "Submitter" is not always the same thing.

• Identifiers – Need persistent identification scheme for more kinds of objects than just content.
• All objects mentioned in the history record must be referred to by persistent and globally unique (or obviously context-dependent) identifiers.
• Required so we can merge histories from different DSpace archives.
• Identifiers that will not always be dereferenceable (e.g. EPerson) ought to include some descriptive metadata.
• Some objects in the DSpace data model, e.g. EPerson, and the archive itself, don’t yet have a globally unique identification scheme.
• We need a persistent identifier for each archive. Handle prefixes do not map 1:1 onto archives. Perhaps a Handle for the archive itself.
• See ‘info:` URIs: http://info-uri.info/  

• Event Monitoring API – New mechanism to add to DSpace core code.
  • Like Rob’s idea of publish/subscribe model for events that change the data model.
  • Generalized interface for many purposes: history, logging, preservation activities like AIP refreshing, etc.
  • Invoked after every atomic change to object model (content model) of the archive.
  • Includes all variable data in History schema:
    • Type of the action (creation, modification, etc).
    • Date and Time it occurred.
    • ‘Context’ object (includes authenticated person, transaction, application identity, etc)
    • Outcome - details of changes made.
    • Additional DSpace objects acted upon, e.g. Collection that Item was added to.

• Software Identification – If the history includes the name and version of "software modules", then we need to add such a notion to DSpace:
  • Naming convention for "applications" (any client of the DSpace object API).
  • Add an API to let application declare its name (e.g. in ‘Context’).
  • Perhaps extend remote interfaces like LNI to let its client application identify itself, rather than appearing as "LNI".
  • Examples: “WebUI 1.4”, “LNI 1.4.1”, “ItemImporter 1.4.1”.
  • See ‘info:` URIs: http://info-uri.info/  

• Digital Signature Infrastructure – Required if we need to guarantee authenticity of records.
  • Each DSpace archive must have a digital identity, such as a digital identity certificate like X.509, signed by a trusted authority.
  • Do we want to get involved in PKI or trust an outside provider?
  • Any proposal must be audited by a qualified expert in security and cryptography.

Next Steps

1. Review by MIT DLRG for sanity check and initial feedback.
2. Post publically to DSpace wiki and invite development community to review it.
3. Integrate comments and changes.
4. Use this information to drive the design of a new history system.