Preservation and Archiving Special Interest Group Spring Meeting
San Francisco, 27-29 May 2008

Preservation Is Not A Location

Stephen Abrams
John Kunze
California Digital Library
Programmatic focus

• In order to be effective over any interesting period of time, preservation needs to be considered from a board programmatic orientation, as opposed to a more narrow project or systems focus

• Many other digital library services (e.g. high-volume end-user access) can benefit from features traditionally discussed only in a preservation context
Desiderata

• “Entities should not be multiplied beyond necessity”  
  – William of Occam

• “The supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience”  
  – Albert Einstein

• How simple can a preservation environment be and still be effective?
Digital preservation

• A set of intentions, activities, and (hopefully) outcomes aimed at the usability of authentic digital objects over time

• Intentions can be articulated in terms of desirable object- and service-centric values

• Activities can be articulated in terms of strategies designed to foster those values
## Object-centric values and strategies

<table>
<thead>
<tr>
<th>Value</th>
<th>Justification</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>To distinguish an object from all others</td>
<td>Persistent naming</td>
</tr>
<tr>
<td>Viability</td>
<td>To recover an object from its medium</td>
<td>Redundancy, heterogeneity, media refresh</td>
</tr>
<tr>
<td>Fixity</td>
<td>To ensure that an object is unchanged from its accepted state</td>
<td>Redundancy, error correcting codes, message digests</td>
</tr>
<tr>
<td>Authenticity</td>
<td>To ensure that an object is what it purports to be</td>
<td>Cryptographically-secure signatures</td>
</tr>
<tr>
<td>Ontology</td>
<td>To understand the significant nature of an object</td>
<td>Syntactic, semantic, and pragmatic characterization</td>
</tr>
<tr>
<td>Visibility</td>
<td>To enable patrons to find objects of interest</td>
<td>Public discovery</td>
</tr>
<tr>
<td>Utility</td>
<td>To expose the underlying information content of an object</td>
<td>Behavior-rich delivery</td>
</tr>
<tr>
<td>Appraisement</td>
<td>To understand the consequences of the passage of time</td>
<td>Analysis and assessment</td>
</tr>
<tr>
<td>Timeliness</td>
<td>To know when a preservation value is threatened</td>
<td>Technology watch</td>
</tr>
</tbody>
</table>
Service-centric values and strategies

<table>
<thead>
<tr>
<th>Value</th>
<th>Justification</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>To provide access at the time of a patron’s choosing</td>
<td>Redundancy, automated failover</td>
</tr>
<tr>
<td>Responsivity</td>
<td>To provide appropriate throughput in servicing requests</td>
<td>Redundancy, automated load balancing</td>
</tr>
<tr>
<td>Security</td>
<td>To enforce appropriate use of systems and content</td>
<td>Identity management, access control lists</td>
</tr>
<tr>
<td>Sustainability</td>
<td>To ensure ongoing access and use</td>
<td>Institutional commitment, financial cost-recovery, staff retention and education</td>
</tr>
</tbody>
</table>

- These strategies can be codified in terms of abstract services, which in turn can be implemented through
  - Human activities
  - Automated systems
Preservation objects
Preservation objects
Preservation objects
Storage substrate

• Ideally, storage is provided as a ubiquitous commodity at a range of functional service levels (and, presumably, price points):

  – Coherence : Unstructured ⇔ Structured
  – Resilience : Unreliable ⇔ Dependable
  – Permanence : Transient ⇔ Persistent
  – Performance : Slow ⇔ Fast

• Object components should be assigned to the least functional, lowest cost storage than minimally meets their requirements at a point in time
Preservation services

- A set of simple, orthogonal services whose invocation can be requested or scheduled
  - Ingest
    - Characterization
    - Normalization
    - Enrichment
    - Naming
  - Archival storage
    - Fixity
    - Replication
  - Data Management
    - Indexing
    - Querying
    - Logging
    - Reporting
  - Access
    - Discovery
    - Request
    - Packaging
    - Delivery
Technological invariance

• 1988
  – FTP
  – POSIX command shell and file system interface
  – RDMS / SQL

• 2028?
  – HTTP
  – URI
  – XML

• Due to their inherent abstract nature, protocols and interfaces last longer than systems
Preservation services

• Easily deployed in sufficient number and location to meet demand
Preservation begins far upstream

• Born-archival objects
  – Easy, fast format characterization in hands of Producers
  – Early, cheap generation of identifiers suitable for persistent reference
  – Better metadata via heuristics (eg, GNU autoconf) and prompting

• Sharing – “preservation implies more than one location”
  – Common object substrate: Pairtree, Eflat
  – Object exchange: BagIt, GrabIt
  – Spreading the risk: format desiccation and diversification

• Transparency and simplicity help us focus on funding and cooperation
Easy, fast object characterization

- Producer-side validation
  - Push characterization operations as far up-stream as possible
  - Early detection of anomalous or problematic data facilitates efficient remediation
  - Requires combination of tools and education
Early, cheap preservation-ready identifiers

- Identifiers with decent chances of persistent reference are usually assigned very late
  - Strangely, objects are often renamed at peak of valuation
- Can we freely give out preservation-ready IDs, even if only a fraction ever return attached to “valued” objects?
  - Lightly-controlled “minting” services, e.g. like tinyurl
  - Requires combination of tools and education
Better object description via heuristics

• Metadata won’t go away, but collecting it is a pain
  – Who, what, when, where, …, <technical metadata>
  – Some metadata is easy to generate, some is not worth generating

• Tool to generate good guesses with user prompt to correct
  – Think GNU “autoconf”, to poke around a computer to develop very sophisticated system metadata guesses
  – Could be applied at multiple life-cycle stages
Preservation implies more than one location

Pairtree: thinnest possible smear on top of a file system to make an object system

- Platform-independent file hierarchy that factors the scarily-hard repository into
  - Easy, powerful names
  - The stuff that’s merely hard
- Common substrate for simple or complex access and preservation systems
What a pairtree gives you

• A file system hierarchy mapping an ID string to a unique object directory using pairs of characters
  
  
  \[ \text{abcdefg} \Rightarrow \text{ab/cd/ef/g/} \]
  
  – There there: all the object’s files and nothing but the object’s files

• Import a pairtree and, knowing *nothing* about objects’ nature, reliably
  
  – Enumerate all objects and their identifiers
  – Produce any object by requested ID
  – Maintain and back it up with ordinary OS tools
  – Rebuild the collection in case of database corruption simply by walking the filesystem
Where pairtree leaves off

Inside object is another story, such as, “eflat”

```bash
object/
  |  meta.txt
  |  files.txt
  |  data/
  |  v001/ . . . v004/
  |  meta/
  |  m001/ . . . . . . . m039/
  |  annotations/
  |  audits/
  |  config/
  |  . .
  |  pairtree...
```
Object exchange with BagIt and GrabIt

• Need: to move *lots* of files from CDL to Library of Congress
  – BagIt file package format
  – GrabIt exchange protocol

• Informed by lessons from
  – AIHT transfer test
  – ARC file format, and
  – “Enclose and Deposit” (Tabata and Sugimoto, IWAW 2005)
BagIt file package format

• A hierarchical file exchange package suitable for…
  – Generic content (no knowledge of bag payload required)
  – Disk- or network-based transfer
  – Possible bag return on a “rainy day”
  – Optional packing metadata & checksums

• Spec at http://www.ietf.org/internet-drafts/draft-kunze-bagit-01.txt
BagIt “bag” structure

- A “bag” reserves just enough file names to permit the safe enclosure of manifest, checksums, “tag” info, arbitrary payload, and optional “holes” for space/time efficiency

  `<bag_dir>/`
  
  | manifest-md5.txt | complete file list + checksums
  | bagit.txt | declares this to be a “bag”
  | package-info.txt | optional packing metadata
  | fetch.txt | optional URL list completing bag

  `--- data/`
  
  | . . . | arbitrary payload files
GrabIt package exchange protocol

• Bag it, tag it, but don’t ship it…
  – Instead, grab it, since a push is really a pull

• GrabIt is intended for moving large batches
  – When you have lots of bags (sent as “tarballs”)
  – Or lots of other file sets, such as ARC containers

• Beats the tedium and error rate in emailing URL lists
Spreading the risk: data desiccation

- Generation of long-lived, perhaps feature-poor derivatives
  - Store derivatives along with originals
  - If original fails, desiccated version has better chance of survival and retains most of the original’s value
  - Should generated derivative close to height of format popularity, when implicit knowledge is transferred
The Big Risks

• Transparency and simplicity help us focus, cooperate, and regenerate

• Must not be distracted from the Big Risks:
  – Political or financial loss (e.g., bankruptcy)
  – War, social upheaval, natural disaster
  – Power outage, disk failure, human error