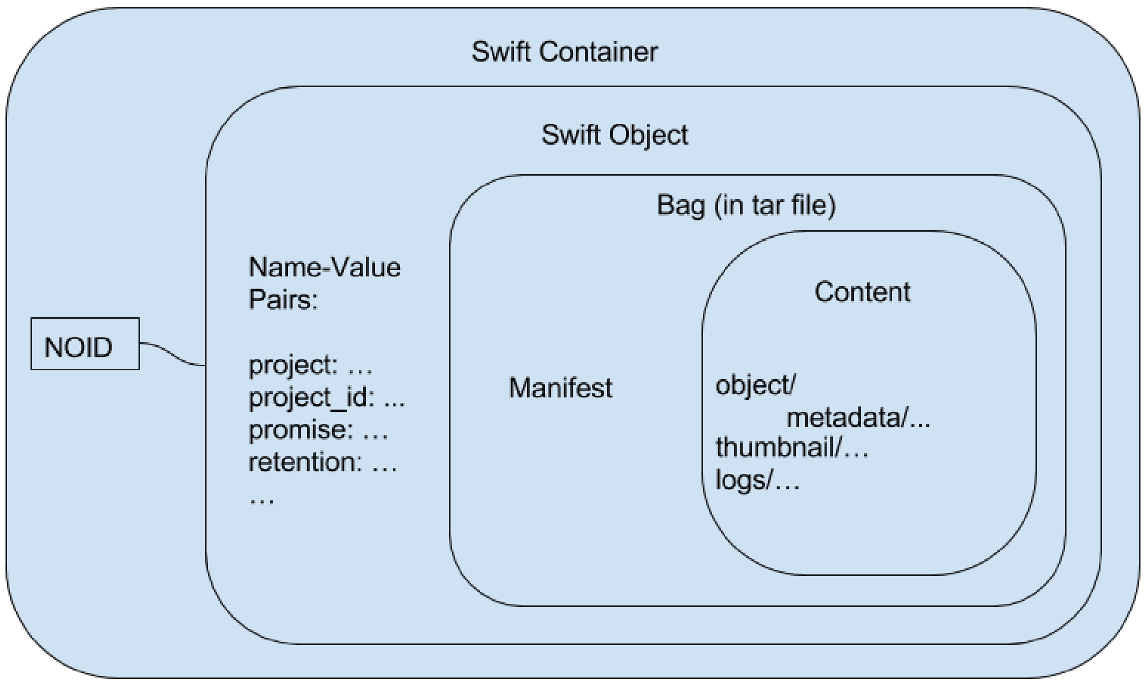
Lightweight AIP Structure and Workflow

Draft 2017-04-24 pbinkley



Demo script

https://gist.github.com/pbinkley/060ed60fb580c8c26324d07048ea1518

Populating the AIP

A lightweight AIP will contain three directories: object, thumbnail, and logs. Metadata is stored in object/metadata. This structure is modelled on Archivematica's AIP structure.

This describes an AIP for an ERA item, drawn from a Fedora object. For these items, all of these files are required.

- Metadata: normally an rdf dump of the related Fedora objects. May include other structured metadata when available. Files:
    - Main Fedora object as objects/metadata/object_metadata.n3 (from $FEDORA_URL/rest/prod/$FEDORA_PATH)

    - Datastream metadata as objects/metadata/content_fcr_metadata.n3 (from $FEDORA_OBJECT/content/fcr:metadata)
    - Content versions as objects/metadata/content_versions.n3 (from objects, discovered by Solr query)$FEDORA_OBJECT/content/fcr:versions)
    - Permissions as objects/metadata/permission_<uuid>.n3 (from permission
- Datastreams: normally the file associated with an ERA object (in current pre-PCDM HydraNorth, where an item can have only one file). May include arbitrary subdirectory structure if required by the content (e.g. for non-ERA digitized items) File:
    - Content datastream (e.g. PDF) as objects/<filename> (from $FEDORA_OBJECT/content, saved with the filename from <model:downloadFilename> in object_metadata.n3)
    - Thumbnail datastream as thumbnails/thumbnail (from $FEDORA_OBJECT/thumbnail)
- Logs: any logs from the creation of the AIP
    - Characterization (PREMIS) as logs/content_characterization.xml (from $FEDORA_OBJECT/characterization)
    - Fedora fixity report: logs/content_fixity_report.n3 (from $FEDORA_OBJECT/content/fcr:fixity)
    - Log of creation process: logs/aipcreation.log (stdout from ingest script)

A typical AIP will therefore look like this:

- ./logs
    - /aipcreation.log
    - /content_characterization.xml
    - /content_fixity_report.n3
- ./objects
    - /metadata
        - /content_fcr_metadata.n3
        - /content_versions.n3
        - /object_metadata.n3
        - /permission_93195c08-4770-4f46-80d9-b03321086b7d.n3
        - /permission_a135db45-600f-4bdc-8f6c-568e4084c736.n3
        - /permission_e0cc8677-ed61-42ce-9491-8906a9a532dc.n3
    - /NSERC Roundtable Talk - Dr. Funk.pdf
- ./thumbnails
    - /thumbnail

For non-Fedora objects, the same structure is used but the only required file is the content file in the objects directory, and a metadata file in objects/metadata; other files may or may not be present, and may or may not be in n3 format. (We will establish the rules in consultation with the Metadata Team as we move beyond the ERA content).

Packaging

This content will be bagged, producing a Bagit manifest and other artifacts. The contents of the AIP will end up in the “data” directory of the bag. The bag directory will be tarred (uncompressed) and pushed into Swift using Swift’s REST API. The bag file will be named “aip.tar”. A checksum for the bag file will be generated and used to validate the upload Swift if possible, or we’ll calculate a checksum on the fly as the tar is uploaded and compare it with Swift’s checksum at the end.

Ingest to Swift

Swift container: “ERA”
Swift path: <NOID>

The identifier in Swift will normally be the NOID generated in ERA. It may be some other identifier for non-ERA content (an identifier used in that project, or a UUID if nothing else is available). Swift X-Object-Meta name-value pairs will be populated:

- project: normally “ERA” (this may become redundant if we always use a container per project, but we haven’t determined that yet)
- Project_id: (i.e. the id of this object within the project) - normally the NOID
- promise: a label for the level of preservation promised
    - This will ultimately cover various levels of preservation as defined in the UAL Digital Preservation Plan (gold, silver, bronze). For lightweight AIPS this will always be “bronze”.
- retention: (optional) a timestamp indicating when some action expressed by the promise will be taken
    - Not applicable to ERA content, but for other content we might use it to trigger a review after 10 years etc.
- depositor: (optional) name of a non-UAL depositor
- AIPversion: 1.0

The Swift container will have versioning enabled, so that all versions of the object are preserved. The consequences for storage usage will be evaluated after we’ve populated Swift, and we may institute policies for version retention.

Return flow of information (Swift to Fedora):

After ingest into Swift, we will capture information about the Swift object (using Swift’s API) and store it in the ERA Fedora object (using Fedora’s API). This information should be stored in a way that meshes with Fedora’s audit trail (determining the appropriate place to store these properties and the property names is a TODO). This information would include:

- Swift path
- Ingest timestamp

- Swift checksum (object level, i.e. for the tar)

Validation Scenarios

- Pre-fetch
  - Validate fixity using Fedora API ($FEDORA_OBJECT/content/fcr:fixity)
  - if this fails, the object is already corrupt in Fedora, so it should not be ingested until the problem has been resolved
- Pre-bag (after the content has been gathered)
  - Validate checksums for datastreams against checksums in the Fedora metadata and in the characterization datastream (i.e. the FITS output)
    - characterization: /fits/fileinfo/md5checksum
    - content_fcr_metadata: <.../content> fedora:digest <urn:sha1:xxxxx>
  - If this fails, the object was corrupted in the retrieval process (and may have been corrupt in Fedora). Do not ingest, resolve manually for now.
- Post-bag
  - Validate the bag
  - If this fails, the bagging process corrupted the object. Resolve manually for now.
- On Swift ingest
  - Provide the bag-level checksum with the PUT into Swift, so that Swift will reject it if it doesn't match
  - If this fails, either Swift had a problem or the calculated checksum was corrupt. Resolve manually for now.
- Periodic Post-ingest Audit
  - Verify the bag-level checksum stored in Fedora against the Swift object
  - Extract the Swift object and validate the bag
  - Verify Fedora's datastream checksums against the extracted checksums

Light-weight AIP Creation
(This is partially obsolete but kept for reference)

Micro-Service	**Description**
Initiate Transfer	Once the transfer folder has all the its digital objects and has been formatted for processing, a user or "system" initiates the AIP creation process
Verify transfer	Verifies that the transfer conforms to the folder structure required for processing. The structure should be ...
Validate Integrity	Validate fixity using Fedora API. Validate checksums for datastreams against checksums in the Fedora metadata and in the characterization datastream
Assign NOID	Assign a unique NOID to the folder (if necessary)

Extract Packages	Extracts objects from any zipped files or other packages
Scan for Viruses (not applicable to ERA)	Uses an anti-virus software to scan for viruses, if found move the package to
Characterize and extract metadata	Identifies and validates formats and extracts object metadata using the File Information Tool Set (FITS). Adds output to the PREMIS metadata.
Add rights statement	Add rights statement to the PREMIS
Assign checksums and filesizes	assign md5, sha1 and sha2 to each file in the submission.
Prepare AIP	Packages the SIP into an AIP using BagIt
Compress AIP	Losslessly compresses the AIP for storage