Part 4: Integrating The Packager

As the software which interacts with the Packager, the Intake services need to be updated to apply the new file layout to any incoming data in order to support new functionality.

Tokenization DECISION

As one of the processes which creates ACE Tokens for our collections, any changes to how we deal with when we create tokens and over what files would occur in intake. This is a good time to think about important questions regarding our current practices: **what files do we need to create tokens for and is it wasteful to create tokens for packaging metadata?**

In general the payload files are what we care the most about being synchronized between files, as the metadata for the file layout is used for verification on transfer, not for long term preservation. Therefore we might want to consider a workflow where we only create ACE Tokens for the payload files of a package, and let each replication service validate the metadata.

Additionally, if the ACE Tokens are embedded in the collection there is a question of if we need to store them in the Ingest database. Currently they are stored in the database so that they can be distributed to each Chronopolis Node.

OCFL Packaging

The workflows defined will be specifically for the OTM Bridge but are general enough that they will need to be applied to other Chronopolis Intake workflows

New Packages

The workflow for creating a new OCFL Object should be very similar to the current flow, with the exception that we are now applying a different packaging format. There are additional steps which may be taken depending on what we decide is included.

Workflow

1. Query OTM Bridge service for a deposit waiting to be ingested
    a. List Deposits
2. Query Ingest to check for existence of the OCFL Object
    a. For this workflow, assume we see that it does not exist
3. Use OCFL Packager to create a new OCFL Storage Root and OCFL Object
    a. Handles creation of the namaste, `inventory.json`, version directory, etc
    b. Handles moving files into the `content` directory
    c. Checks `fixity` for files during movement of data
4. Create optional files
    a. Generate ACE Tokens and put them in a Token Store
    b. Generate logging information if needed
5. Finalize the OCFL Object
    a. Packager generates `inventory.json.{alg}`
6. Register the OCFL Object with the Ingest Server
    a. Load Files
    b. Load Fixity
    c. Load ACE Tokens
        i. If these are embedded in the package, is it still needed?
7. When the OCFL Object is marked as PRESERVED
    a. Create Audit Events for Ingest and Replication

Applying Versions

Depending on how much work we want to do, the workflow for applying a new payload to a package has additional steps which must be taken. In addition, this process requires more communication as data will need to be re-staged so that it can be modified.

Workflow

1. Query Bridge service for a preservation package waiting to be ingested
    a. List Deposits
2. Query Ingest to check for existence of the OCFL Object
    a. For this workflow, we assume that the Ingest tells us that a package does exists
3. Request staging of the OCFL Object
    a. Ideally should go through the same Chronopolis node that Ingested the package to begin with
    b. Could be a workflow of its own, potentially through a staging service
    c. We should aim to stage the minimum amount of data needed
        i. The payload of the OCFL Object should not be necessary as we aren't computing differences on files and have access to the manifest through the `inventory.json`
4. Modify the preservation OCFL Object
    a. Validate fixity
    b. Add new version timestamp, number, files
    c. Move the payload files in to place
    d. Update package metadata for versioning
5. Create optional files

    a. Generate ACE Tokens for new files
    b. Generate logging information if needed
6. Finalize the OCFL Object
    a. Regenerate the `inventory.json.{alg}`
7. Update the OCFL Object in the Ingest Server
    a. Create new version of the Collection
        i. Registers new fixity for files
        ii. Registers new tokens for files
    b. Files which do not change are not re-distributed
8. When the OCFL Object is marked as `PRESERVED`
    a. Create Audit Events for Ingest and Replication

Deleting Versions

Workflow Decisions DECISION

Before implementing a workflow for deleting versions and version data from a collection, we first need to decide how that will occur and what implications that has on the system.

Repackaging Considerations

In OCFL, when removing a file it is recommended to "create a new object that excludes the offending file, with a revised version history taking this into account." (OCFL Implementation Note - File Purging). We should abide by this if possible, in order to keep our packages consistent with the OCFL best practices. When creating a new package, there is the option to either remove the file entirely, or provide a `stub` which replaces the file but keeps the identifier.

It is conceivable that repackaging can be done solely on the `inventory.json` and `inventory.json.{alg}` files, as well as overriding the of the deleted file so that a `stub` can be transferred throughout the Chronopolis network.

Distributed Repackaged Collections

Once a collection is repacked, it will need to be redistributed throughout Chronopolis. As the changes to the OCFL Object will modify every `inventory.json` from when the deleted file was introduced, we will want to look at the best way to distribute these changes throughout the system.

Overwrites

By overwriting files, we can transfer less overall data to the nodes in Chronopolis. However, we would need to make modifications to each `inventory.json` and `inventory.json.{alg}` in the Ingest Server as well as each ACE AM.

Deprecation and Redistribute

This is similar to the current Chronopolis workflow in which collections are marked as `DEPRECATED` and removed from the ACE AM instances at each site in Chronopolis. By doing this, we can re-ingest a collection and distribute it as one whole operation. This would need to ensure that all versions of the collection are still available in the Ingest Server, and that replications are able to grab the entire OCFL Object.