Part 1: Ingest

The Ingest Server holds information about al collections in the Chronopolis Network. As such, it will need to at least have some updates in order to accommodate a new file layout, as well as additional updates which could be added for quality of life when requesting information about collections.

Database Migrations

Bag Table

The Ingest Server currently assumes that every collection coming in is using a BagIt format. Regardless of what packaging format we choose, we should limit assumptions made by the Ingest Server and use a more generic name for data being preserved. We should align nomenclature of the Ingest Server with ACE-AM and refer to our grouped data as Collections. Additional information will sit alongside to describe the packaging format used, which would allow for migration in the future if a new standard comes along which is better than what we apply. In addition, this information could be used in order for our Intake and Restore services to return data back to a depositor.

Overview of Changes

- Create and populate a `collection_layout` table pre-populated with known layouts
  - BagIt
  - OCFL
- Add an additional `collection_layout_id` column on the bag table for the collection layout
  - Migrate current collections to have the BagIt id
  - Set `collection_layout_id` column to the id for BagIt
- Use unified naming for collections
  - Alter the name of `bag` to `collection`
  - Alter the sequence `bag_id_seq` to the `collection_id_seq`
  - Alter the table name for `bag_distribution` to `collection_distribution`
  - Alter the sequence `bag_distribution_id_seq` to `collection_distribution_id_seq`
- JPA Entity Updates (`rest-entities`)
  - `Bag`
    - Add `CollectionLayout`
    - Rename to `Collection`
  - `BagDistribution`
    - Rename to `CollectionDistribution`
  - `BagDsitributionStatus`
    - Rename to `DistributionStatus`
- JPA Entity Serializer Updates (`rest-entities`)
  - `BagSerializer` rename to `CollectionSerializer`

Versioning Support

There will be a need to have some level of versioning in the database in order to better support a collection layout which has versioning as well. We will need information in a collection about what version it is on as well as the files required for the collection

Collection Level

The primary table which is used to track collections will need to be updated to contain a reference to the latest version. The `collection_version` table does not need to be large and should reflect the same information as in the OCFL inventory file: id, created, and number. There are additional fields in the OCFL inventory for storing a message and the user who created the version, which could also be added to the table. In general these tables can be thought of as a cache of what is in Chronopolis, and should be able to be rebuilt by iterating through what is on disk if necessary.

New Table: collection_version

	Column	Type	Comment
+	id	bigint	Primary key for the table
+	collection_id	bigint	Id for joining version information to a collection
+	number	bigint	Natural number which can be incremented for determining the numbered version of a collection
+	created_at	datetime	Date at which the collection version was created

File Level

The Ingest Server currently tracks Files and associated metadata in the form of Fixity and ACE Tokens. If files are to be updated, then they also need some type of version information. It is likely that both the Fixity and ACE Token tables will need some information as to what version of a File they are pointing to, and likewise a File will need to know what the latest version is. As we do not update Files separately from the Collections which they belong to, piggy backing off of the `collection_version` will likely be the best choice to make here. In order to do this, we can add additional columns to the fixity and `ace_token` tables for the `collection_version_id`.

Updated Table: fixity table updates (might remove uniq idx)

	Column	Type	Comment
+	collection_version_id	bigint	Indicate the collection version which this fixity (and by proxy file) belongs to
+	(file_id, collection_version_id)	Unique index	Index and indicate uniqueness of files to the collection_id

Updated Table: ace_token (might remove uniq idx)

	Column	Type	Comment
+	collection_version_id	bigint	Indicate the collection_version which this ace_token belongs to
+	(file_id, collection_version_id)	Unique index	Index and indicate uniqueness of files to the collection_id

Normalization Opportunities

The fixity table contains columns which could be broken out into separate tables: algorithm and value. Both of these columns could be referenced by an id and joined on a separate table. The algorithm column specifically should belong to a `supported_algorithms` table from which we store the message digest algorithms which we support. For the value column, a `fixity_value` table could be created which would reduce duplication of fixity values stored. This is less of a concern than the algorithms, but still something to consider.

Changes

- Collection version database migration
  - Create `collection_version` table with required columns
  - Add `collection_version` for each collection which defaults to version 1 and uses the Collection's `created_at` for its `created_at`
- Collection version database entities
  - Create `CollectionVersion` class under rest-entities to create a JPA Entity
  - Update `Collection` class and add the relationship with the `CollectionVersion`
- Collection version models
  - Create `CollectionVersion` class under rest-models
  - Expand `Collection` class to include current version information
- File level versioning database migration
  - Update `fixity` table to add a `collection_version_id` column
  - Update `ace_token` table to add a `collection_version_id` column
  - Set default `collection_version_id` to be the `collection_version` for the `Collection` they belong to
  - Create `unique_index` for `fixity` and `ace_token` tables on `(file_id, collection_version_id)`
- File level entity updates
  - Update `Fixity` class under rest-entities to include its relationship to the `CollectionVersion`
  - Update `AceToken` class under rest-entities to include its relationship to the `CollectionVersion`
- File level query updates
  - Under ingest-rest, ensure queries use the current version of a collection when looking for files

API Updates

In order to facilitate versioning of content, the REST API on the Ingest Server will need to be expanded. This will involve creating an endpoint to create Versions on collections, querying versions for collections, and possibly retrieving File data for Versions.

Models

Before digging into the endpoints, it is important to look at the models which we will need to add to support Versioning.

CollectionVersion

The same information we receive from the database:

- `id`
- `version_number`
- `created_at`
- `collection_id`

VersionManifest

A plain text listing of Files and their Fixity values. Originally was going to be just the Files, but since metadata files can change it is important to include the Fixity as well.

Example

```
db84dd4fb5dfc0cef5f0509e9e910ee6f416c2df  data/manifest.txt
d96f61bfe6f832dad3a73c09bb177967282c2400  data/content-properties.json
20c59b1614adc782c889301ba6f2bb46e998fab7  data/.collection-snapshot.properties
```

API Model Updates

- Bag
  - Add `layout`
  - Rename to `Collection`
- BagCreate rename to CollectionCreate
- BagStatus rename to CollectionStatus
- API Endpoint Updates
  - BagController (`ingest-rest`)
    - Rename to CollectionController
  - Update route to use collection
  - BagService (`rest-models`)
    - Update routes to use collection

New and Update API Calls

PUT Collection Version

Description

In order to create a version, a `VersionManifest` can be uploaded to an endpoint for a given collection resource, which would save a new set of `Files` and `Fixities` for a given version. As mentioned previously this is essentially a cache of what is on disk, and should be able to be rebuilt from the data in the collection. Since this information is coming from an update on disk, we know where the Version will be able to be located prior to creating the resource in the Ingest Server.

This endpoint could replace the **Create Files** endpoint, which serves to populate a collection with a set of files from a given manifest.

Request/Response

- Request: **`PUT`** `/api/collections/<collection_id>/version/<version_id>`
- Request Parameters:
  - `collection_id` -
  - `version_id` -
- Request Body: `VersionManifest`
- Response Body: Undetermined
- Response Codes:
  - `201 - Created`
  - `400 - Bad  Request`
  - `401 - Unauthorized`
  - `403 - Forbidden`
  - `409 - Version Exists`

GET Current Version

Description

Get the current version for a given collection

Request/Response

- Request: **`GET`** `/api/collections/<collection_id>/version`
- Response Body: `CollectionVersion`
- Response Codes:
  - `200 - Ok`
  - `401 - Unauthorized`
  - `404 - Not Found`

GET Current Manifest

Description

Get the manifest for the current version for a given collection

Request/Response

- Request: **`GET`** `/api/collections/<collection_id>/manifest`
- Response Body: `VersionManifest`
- Response Codes:
  - `200 - Ok`
  - `401 - Unauthorized`
  - `404 - Not Found`

GET Version

Description

Get the current version for a given collection

Request/Response

- Request: **`GET`** `/api/collections/<collection_id>/version/<version_id>`
- Response Body: `CollectionVersion`
- Response Codes:
    - `200 - Ok`
    - `401 - Unauthorized`
    - `404 - Not Found`

GET Version Manifest

Description

Get the manifest for a given collection Version

Request/Response

- Request: **`GET`** `/api/collections/<collection_id>/version/<version_id>/manifest`
- Response Body: `VersionManifest`
- Response Codes:
    - `200 - Ok`
    - `401 - Unauthorized`
    - `404 - Not Found`

Deprecated Endpoints

Some endpoints can be superseded by the versioning endpoints. These include:

GET /api/bags/<bag_id>/download

POST /api/bags/<bag_id>/files