APPENDIX H - All About Tuque

Connecting to a Fedora Repository

To connect directly to a Fedora Repository using the tuque library:

```
$connection = new RepositoryConnection($fedoraUrl, $username, $password);
$connection->reuseConnection = TRUE;
$repository = new FedoraRepository(
        new FedoraApi($connection),
        new SimpleCache());
```

The IslandoraTuque class provides a simple constructor for Islandora users.

```
D7 module_load_include('inc', 'islandora', 'includes/IslandoraTuque');
D6 module_load_include('inc', 'islandora', 'includes/tuque');
$repository= new IslandoraTuque();
```

All interaction with Fedora can now take place through the $repository object. D7 There are some wrapper functions that handle some errors and fire some hooks in islandora.module.

To build a new object:

Create array of ContentModels for the object. This will normally be a single element array:

```
$content_models = array(array('pid' => 'islandora:collectionCModel'));
```

Identify the namespace for the new pid, and identify the collection the object is to be a member of. Tuque will retrieve the next available PID in that namespace.

```
$namespace = 'test';
$collection_pid = 'islandora:root';
```

Create the object using the factory method.

```
module_load_include('inc', 'islandora', '/includes/islandora.ingest');
$fedora_object = islandora_ingest_get_object($content_models, $collection_pid,
'isMemberOf', $namespace);
```

or create the object directly with tuque:

```
$fedora_object = $repository->constructObject($namespace); // allow fedora to generate
a PID
```

or with a specified PID:

```
$fedora_object = $repository->constructObject($pid); // create an object with the given
PID
```

Label the object

```
$fedora_object->label = "my new object";
```

Set the owner

```
$fedora_object->owner = $username;
```

Build datastream

```
$datastream_id = "TN";
$new_datastream = $fedora_object->constructDatastream($datastream_id);
```

or

```
$datastream_id = "MODS";
$controlGroup = "X";
$new_datastream = $fedora_object->constructDatastream($datastream_id, $controlGroup);
```

Set label, mimetype, and content as a minimum. Content may come from url, string, or file

```
$new_datastream->label = 'MYDSID';
$new_datastream->mimetype = 'something/something';
$new_datastream->setContentFromUrl(URL_TO_CONTENT);
```

or

```
$new_datastream->setContentFromFile(PATH_TO_CONTENT);
```

or

```
$new_datastream->setContentFromString("content");
```

Add datastream to object

```
$fedora_object->ingestDatastream($new_datastream);
```

Manipulate object's RELS-EXT

```
$fedora_object->relationships->remove(FEDORA_MODEL_URI, 'hasModel', 'islandora:
collectionCModel');
$fedora_object->relationships->add(FEDORA_MODEL_URI, 'hasModel', 'islandora:
imageCModel');
```

Ingest object into fedora repository

```
$new_fedora_object = islandora_ingest_add_object($fedora_object);
```

or ingest with existing repository object

```
$repository->ingestObject($fedora_object);
```

Working with existing objects.

Create object to access and manipulate existing Fedora object.

```
$pid = "test:1";
$fedora_object = islandora_object_load($pid);
```

or

```
$fedora_object = $repository->getObject($pid);
```

Object evaluates to FALSE if the Object doesn’t exist.

```
if (!$fedora_object) {
drupal_set_message("Fedora Object isn't in the repo!");
}
```

Purge Object from repository

```
$fedora_object->repository->purgeObject($pid);
```

or

```
$repository->purgeObject($pid);
```

Delete Object from repository

```
$fedora_object->delete
```

Accessing RELS_EXT

Returns an array of associative arrays representing all relationships.

```
$rels = $fedora_object->relationships->get()
```

Each array has two keys, each pointing to its own associative array of values.

```
[0] = Array
        (
          [predicate] => Array
                 (
                        [value] => hasModel
                        [alias] => fedora-model
                        [namespace] => info:fedora/fedora-system:def/model#
                 )
          [object] => Array
                 (
                        [literal] =>
                        [value] => islandora:collectionCModel
                 )
        )
[1] => Array......
```

Returns array of relationships filtered by namespace and predicate

```
$rels = $fedora_object->relationships->get('info:fedora/fedora-system:def/model#',
'hasModel' );
```

Datastreams

Get all datastreams as an array

```
$datastreams = $fedora_object->repository->api->a->listDatastreams($pid);
```

Get individual datastream by DSID

```
$datastream = $fedora_object['dsid'];
```

Get properties of Datastream.

```
$pid = $datastream->id;
```

All datastream properties. Not case sensitive.

- label
- controlGroup
- versionable
- state
- mimetype
- format
- size
- checksum
- checksumType
- createdDate
- content
- url
- location
- logMessage

Iterate through datastreams in an object.

```
foreach($fedora_object as $datastream){
 // access individual datastreams.
}
```

Properties can be accessed or mutated.

```
$old_mime = $datastream->mimeType;
$datastream->mimeType = "new/mimetype";
$datastream->content = file_get_contents('http://myexample.com/sample.jpg'));
```

Creating or updating a datastream that may or may not exist.

```
// Create DS or grab it.
if (!isset($fedora_object["JP2"])) {
         $jp2_ds = $fedora_object->constructDatastream('JP2', 'M');
 }
        else { $jp2_ds = $fedora_object["JP2"];
 } $jp2_ds->label = 'Derived display JP2.';
$jp2_ds->mimeType = 'image/jp2';
 // Don't copy the file. $jp2_ds->setContentFromFile($jp2_file, FALSE);
 // May not need to be called if the DS already existed
$fedora_object->ingestDatastream($jp2_ds);
```

Access Resource Index

```
$objects = $fedora_object->repository->ri->itqlQuery($query, 'unlimited', '0'); // for
itql
$objects = $fedora_object->repository->ri->sparqlQuery($query); // for SparQL queries
```