# Release 0.1

## Description

The first pilot release of DuraCloud lays the foundation for storage across underlying providers.

Through either the web-ui or via direct REST calls

- underlying providers may be listed
- spaces may be created/deleted
- content may be uploaded/downloaded/deleted
- metadata may be viewed
- metadata may be modified
    - *modification is fully supported through the REST API*
    - *modification is partially supported through the web-ui*

## Deliverables

1. Duradmin instance URL
    - NYPL
        - http://nypl.duracloud.org:8080/duradmin
        - http://nypl.duracloud.org:8080/durastore
    - BHL
        - http://bhl.duracloud.org:8080/duradmin
        - http://bhl.duracloud.org:8080/durastore
    - WGBH
        - http://wgbh.duracloud.org:8080/duradmin
        - http://wgbh.duracloud.org:8080/durastore
2. REST API documentation
3. Store Client - Includes storeclient.jar and all dependent jars as well as Javadocs
4. Simple walk-through instructions for DurAdmin UI
5. Improvement/Bug tracker
6. Bulk upload
    a. load instructions
    b. load software
        - bagit-3.1-bin.zip (and spec)
        - Copied bag verifier
    c. 3 2TB hard drives (NYPL)

# Drive Load Instructions for Bulk Ingest

## Description

The intent of this document is to detail the process for verifying the integrity of content that is copied onto hard drives to be shipped to a storage provider for bulk ingest.

At a high level, the basic approach is to generate a manifest of md5/fileId pairs while the content is still on the source network and again after the content has been copied to the drive.
These two manifests are then compared to ensure no corruption or file loss occurred in the copy process.

The California Digital Library's BagIt standard and accompanying software will be used to generate the md5 manifests.

The BagIt sw provides for the creation of "bags" in two ways:

1. User points to list of directories containing content to be bagged, and BagIt copies it all over to a user-specified destination directory (operation:create)
2. User points to content that is already in the BagIt-defined directory structure, and BagIt creates the metadata and md5 manifests 'in-place' (operation:baginplace)

| BagIt bag directory structure |
| --- |
| bag/<br>bag/data/<br>bag/data/[duracloudpilot:content]<br><br>*where [duracloudpilot:content] is any number of user content files or nested directories of user content* |

| Complete BagIt bag |
| --- |
| bag/<br>bag/manifest-md5.txt<br>bag/bagit.txt<br>bag/tagmanifest-md5.txt<br>bag/bag-info.txt<br>bag/data/<br>bag/data/[duracloudpilot:content]<br><br>*where the four txt files are generated by the BagIt sw* |

## Approach

Assuming bulk ingest deals with large amounts of data (1TB +), the operation:create approach is probably not reasonable. Additionally, given the simple directory structure defined by BagIt, DuraCloud proposes the operation:baginplace approach with the help of symlinks.

## Details

1. User creates a bag and data directory on network accessible to content

```
$mkdir -p {some-path}/source-bag/data
```

2. User creates symlinks from within data/ directory to content directories

```
$cd source-bag/data
$ln -s {path-to-content-dir0}
$ln -s {path-to-content-dir1}
$ln -s {path-to-content-dirN}
```

3. User runs BagIt process:baginplace over virtual bag to generate metadata and manifests

```
$./bag baginplace {some-path}/source-bag/
```

4. User attaches and formats shipment drive to ext3 (or ext2)

```
$mkfs -t ext3 /dev/{sdb1}   [or whatever steps are appropriate for your env]
$fsck -f -y /dev/{sdb1}     [or whatever...]
```

5. User mounts formatted drive

```
$mkdir /mnt/dura
$mount /dev/{sdb1} /mnt/dura
```

6. User copies contents referenced under the data/ directory onto the shipment drive

```
$cp -r {path-to-content-dir0} /mnt/dura
$cp -r {path-to-content-dir1} /mnt/dura
$cp -r {path-to-content-dirN} /mnt/dura
```

7. User creates a bag and data directory at the top of the shipment drive

```
$mkdir -p /mnt/dura/duracloud-bag/data
```

8. User creates symlinks from within data/ directory to content directories

```
$cd /mnt/dura/duracloud-bag/data
$ln -s /mnt/dura/{path-to-content-dir0}
$ln -s /mnt/dura/{path-to-content-dir1}
$ln -s /mnt/dura/{path-to-content-dirN}
```

9. User runs BagIt process:baginplace over virtual bag to generate metadata and manifests

```
$./bag baginplace /mnt/dura/duracloud-bag/
```

10. User verifies content by manually comparing manifest-md5.txt files, or runs [helper utility|Release 0.1^manifest-verifier.jar]

```
$java -jar manifest-verifier.jar {some-path}/source-bag/manifest-md5.txt
/mnt/dura/duracloud-bag/manifest-md5.txt
```

# DuraCloud REST API (release 0.1a)

## DuraStore

**Purpose: DuraStore is the application through which DuraCloud manages storage. The DuraStore REST API provides access to storage by mediating the underlying storage provider APIs to allow access to multiple cloud storage options through a single API.**

**Store REST Methods**

### Get Stores

- Purpose: Provides a listing of available storage providers accounts (without credentials)
- Request: POST http://host:port/durastore/stores
- Parameters: None
- Response Code: 200 (on success)
- Response Body: XML similar to:

```
<storageProviderAccounts>
  <storageAcct ownerId='0' isPrimary='true'>
    <id>1</id>
    <storageProviderType>AMAZON_S3</storageProviderType>
  </storageAcct>
</storageProviderAccounts>
```

**Space REST Methods**

### Get Spaces

- Purpose: Provides a listing of all of the spaces that a customer has created with associated metadata
- Request: GET http://host:port/durastore/spaces ? (storeID)
- Response Code: 200 (on success)
- Response Body: XML similar to:

```
<spaces>
  <space id="0">
    <space-name>Testing Space</space-name>
    <space-count>65</space-count>
    <space-access>OPEN</space-access>
    <space-created>Mon, 01 Jan 2000 08:00:00 EST</space-created>
    <custom-metadata>Custom Metadata Value</custom-metadata>
  </space>
</spaces>
```

### Get Space

- Purpose: Provides a listing of all of the contents of a space along with space metadata
- Request: GET http://host:port/durastore/spaceID ? (storeID)
- Response Code: 200 (on success)
- Response Body: XML similar to:

```
<space id="0">
  <item>Image 1</item>
  <item>Image 2</item>
</space>
```

- Response Headers: All available space metadata, example:

```
x-dura-meta-space-name=Testing Space
  x-dura-meta-space-count=65
  x-dura-meta-space-access=OPEN
  x-dura-meta-space-created=Mon, 01 Jan 2000 08:00:00 EST
  x-dura-meta-custom-metadata=Custom Metadata Value
```

### Get Space Metadata

- Purpose: Provides all space metadata
- Request: HEAD http://host:port/durastore/spaceID ? (storeID)
- Response Code: 200 (on success)
- Response Headers: Same as for Get space (above)

### Create Space

- Purpose: Creates a new space
- Request: PUT http://host:port/durastore/spaceID ? (storeID)
- Request Headers: Metadata about the space, example:

```
x-dura-meta-space-name=Testing Space
  x-dura-meta-space-access=OPEN
  x-dura-meta-custom-metadata=Custom Metadata Value
```

- Response Code: 201 (on success)
- Response Body: Location of the new space (i.e. the URL used to create the space)

### Set Space Metadata

- Purpose: Updates the metadata associated with a space
- Request: POST http://host:port/durastore/spaceID ? (storeID)
- Request Headers: Same as Create space (above)
- Response Code: 200 (on success)
- Response Body: "Space $spaceID updated successfully" (on success)

### Delete Space

- Purpose: Deletes a space
- Request: DELETE http://host:port/durastore/spaceID ? (storeID)
- Response Code: 200 (on success)

**Content REST Methods**

### Get Content

- Purpose: Retrieves a piece of content along with its metadata
- Request: GET http://host:port/durastore/spaceID/contentID ? (storeID)
- Response Code: 200 (on success)
- Response Body: The content stream
- Response Headers: All available content metadata, example:

```
Content-Type=text/plain
  Content-Length=5732
  Content-MD5=345670923478509747383920 2
  ETag=345670923478509747383920 2
  x-dura-meta-content-name=Testing Content
  x-dura-meta-custom-metadata=Custom Metadata Value
```

### Get Content Metadata

- Purpose: Retrieves the metadata of a piece of content without the content itself
- Request: HEAD http://host:port/durastore/spaceID/contentID ? (storeID)
- Response Code: 200 (on success)
- Response Headers: Same as Get content (above)

### Store Content

- Purpose: Adds a piece of content to the store
- Request: PUT http://host:port/durastore/spaceID/contentID ? (storeID)
- Request Body: Content to be added
- Request Headers: Metadata about the content, example:

```
Content-Type=text/plain
  x-dura-meta-content-name=Testing Content
  x-dura-meta-custom-metadata=Custom Metadata Value
```

- Response Code: 201 (on success)
- Response Body: Location of the new content (i.e. the URL used to create the content)

### Set Content Metadata

- Purpose: Updates the metadata associated with a piece of content
- Request: POST http://host:port/durastore/spaceID/contentID ? (storeID)
- Request Headers: Same as Store content (above)
- Response Code: 200 (on success)
- Response Body: "Content $contentID updated successfully"

### Delete Content

- Purpose: Removes a piece of content from the store
- Request: DELETE http://host:port/durastore/spaceID/contentID ? (storeID)
- Response Code: 200 (on success)
- Response Body: "Content $contentID deleted successfully"

# DurAdmin (v0.1a)

## Introduction to DurAdmin

The DuraCloud Administrator (DurAdmin) is a web interface designed to provide simple interaction with DuraCloud. DurAdmin sits on top of two DuraCloud applications, DuraStore and DuraService, which handle the content storage and the service deployment portions of DuraCloud respectively. In this first alpha release DurAdmin will only be interacting with DuraStore, allowing you to view, update, and delete Spaces and Content.

## Definitions

- **Space** - A top-level container in which content files are stored. Spaces are DuraCloud's single level of file organization. Spaces in DuraCloud are similar in nature to Buckets in Amazon S3 and Containers in the Rackspace Cloud.
- **Content** - A file stored in DuraCloud. Each content item in DuraCloud is stored in a space. The name of a content item can indicate a hierarchical structure (e.g. folder1/folder2/myfile.txt) but DuraCloud treats it simply as a file within a space.

## Using the DurAdmin interface

To use DurAdmin, start by opening a web browser and entering the address provided for your pilot organization. (This should be similar to http://organization.duracloud.org:8080/duradmin/) You will see a welcome page with a small amount of information.

Clicking on the Spaces tab near the top left will bring you to the Spaces page. This page lists the spaces that have been created in your account. Hovering over each space will query for the metadata of the space and populate that information on the page. From this page you can:

- Create a new space
  - To create a space use the Add Space link in the upper left corner of the Spaces page. This will give you the opportunity to enter the space ID and to select the space access. A space can be either OPEN or CLOSED; an OPEN space allows for direct anonymous access to the content in a space through the underlying storage provider; a CLOSED space allows only the space owner to access the content.
- View an existing space
  - View a space by clicking on the space name in the list.
- Add content to an existing space
  - Adding content can be accomplished either on the spaces page or in the space details page. To add content from the spaces page, click on the "Add Content Item" link in the space hover menu.
- Delete a space
  - Delete a space by clicking on the "Remove" link in the hover menu. Be warned, this is a recursive delete, all of the content items contained in the space will be removed as part of this activity. Deleting a space CANNOT be undone.

From the spaces page you can navigate to any one space by clicking on the space name in the list. This will provide you with a listing of all of the content items stored in the space. As on the spaces page, hovering over each content item will retrieve its metadata. From this page you can:

- Change the space access property
  - Change the access property of a space by selecting either the "Open Space" link (available if the space is currently closed) or the "Close Space" link (available if the space is currently open) in the upper left corner.
- Add a content item to the space
  - Add a content item by selecting the "Add Content Item" link in the upper left corner. This allows you to select a file and upload it into your space. By default the name of the file will be used as the file's ID, but you can override that value by filling in the Content ID field. By default the system will attempt to discover the correct MIME type for your file, but you can ensure the correct value by entering it in the MIME Type field.
- View a content item
  - View more details about a content item by selecting the name of the item.
- Download a content item
  - Download a file by selecting the "Download" link in the hover menu.
- Delete a content item
  - Delete a content item by selecting the "Remove" link in the hover menu. Deleting a content item CANNOT be undone.

Selecting the name of a content item provides a detailed view of that item. All of the item's metadata is listed, some of which can be updated. DurAdmin does not currently have the ability to add or edit arbitrary metadata, though this is possible through the DuraStore REST API and through the StoreClient Java library.

# Feedback - Release 0.1a

### Purpose

This page provides a place to collect feedback regarding the DuraCloud 0.1 Alpha release. Below is a listing of issues which exist in the 0.1 Alpha release. Feel free to add to the list as you encounter bugs or see areas which need improvement. Please be as descriptive as possible so that it is clear

### Issues

| Location | Name | Description | Reporter |
|---|---|---|---|
| DurAdmin UI | ID Validation | Provide better validation of space and content IDs | Bill Branan |
| DurAdmin UI | User Metadata | Allow addition/modification of arbitrary metadata on spaces and content through the UI | Bill Branan |
| DurAdmin UI | Thumbnails | Generate appropriate thumbnails for content | Andrew Woods |
| DurAdmin UI | Services | Implement Services page to allow deployment/un-deployment of services | Bill Branan |
| StoreClient/DurAdmin UI | Special Characters | Handle special characters (particularly space characters) as spaceIDs and contentIDs | Bill Branan |
| DurAdmin UI | Delete from Details | Ability to delete content item from content details page | Andrew Woods |