DuraCloud Release 1.0

User Documentation

- Release Notes
- Known Issues
- DuraCloud Features
- DuraCloud Services
- DuraCloud REST API
- DuraCloud Security
- DuraCloud Java Clients
- DuraCloud Sync Tool
- DuraCloud Retrieval Tool
- DuraCloud Chunker Tool
- DuraCloud Administration

Developer Documentation

- Building DuraCloud from Source
- Service Development Guide
- Logging Configuration

DuraCloud Chunker Tool

Introduction

The Chunker Tool is a utility which was created in order to provide a simple way to copy files from a local file system to DuraCloud in a "one-off" manner. Actually, although the common case is to use this tool to copy one or more files to DuraCloud, it may also be run to copy files to another location on the local file system.

Download

Download the Chunker Tool from the Downloads page.

Operational notes

- If you want to jump directly into using the tool, download it from the link above and run the following command

```
java -jar chunk-{version}-driver.jar
```

  The resulting usage statement (detailed below) should be enough to help you get started.

- The Chunker Tool allows you to copy multiple local files and directories into a single space within DuraCloud. The names of the objects which are added to DuraCloud will contain all of the directory elements in the path starting from the first element below the base directory down to the individual file names.

Using the Chunker Tool

- To run the Chunker Tool, open a terminal or command prompt and navigate to the directory where the Chunker Tool is located and run the above command.
- The following options are available when running the Chunker Tool

Short Option	Long Option	Arguments	Description

-a	--add	<f t s>	add content from directory:<f> to space or directory:<t> of maximum chunk size:<s>, where the chunk size must have a unit suffix of K,M, or G — If the -c option is provided, the destination space <t> will be interpreted as the name of a space in the DuraCloud account found at the host:port provided in the -c option, otherwise the destination space will be interpreted as a directory on the local file system.
-c	--cloud-store	<host:port>	use cloud store found at <host>:<port> as content destination
-d	--dir-filter	<l>	limit processed directories to those listed in file-list:<l> — If the -d option is not used, all directories under the base source directory provided in the -a option will be included. The file specified by this option is expected to contain a list of directory names each on there own line. The list is converted to an OrFileFilter from Apache Commons IO
-f	--file-filter	<l>	limit processed files to those listed in file-list:<l> – The file specified by this option is expected to contain a list of file names each on there own line. The list is converted to an OrFileFilter from Apache Commons IO
-g	--generate	<outFile numBytes>	generate test data to <outFile> of <size> bytes — This option does not copy any files, it only generates test data files of the size specified in the give argument.
-i	--ignore-large-files	no args	if this option is set, files over the chunk size specified in the 'add' option will be ignored.
-p	--password	<password>	password of duracloud instance
-u	--username	<username>	username of duracloud instance
-x	--exclude-chunk-md5s	no args	if this option is set, chunk MD5s will NOT be preserved in the manifest — It is expected that this option is rarely used, but in certain situations where the MD5s of the segments of a file that needed to be chunked because the parent file was larger than the limit set in the -a option, not generating these MD5s improves performance.

Creating your own Chunks

If you are interested in creating chunked files in DuraCloud using your own tools, you may do so by adhering to the XML schema used by DuraCloud to create chunks.

Download the Chunker XSD from the Downloads page

Logging Configuration

Introduction

The logging framework used in the DuraCloud application is SLF4J with the LogBack implementation statically bound at runtime. See the LogBack website for a detailed description of the configuration options.

The application also contains bridges for both Log4J and Commons-Logging which translates any underlying, dependency libraries which are configured to write to these frameworks into the SLF4J API. The effect is that all logging is channeled through the SLF4J configuration.

General Usage

- By default, if no configuration file is found by LogBack, the logging level is set to "DEBUG" and the appender is set to "STDOUT"
- When starting any DuraCloud application, a LogBack configuration file may be specified by using the following system variable

```
java -Dlogback.configurationFile={path-to-logging-configuration-file} -jar any-application
```

- Additionally, LogBack will use the file named "logback.xml" found at the top of the classpath for configuration
- An example logback.xml file can be found on the Downloads page

```
1<?xml version="1.0" encoding="UTF-8"?>
2
3<configuration >
4   <!--<configuration debug="true" scan="true">-->
5   <jmxConfigurator/>
6   <property name="LOG_FILENAME" value="/home/duraspace/logs/duracloud-osgi.log" />
7
8   <appender name="DURACLOUD" class="ch.qos.logback.core.rolling.RollingFileAppender">
9     <File>${LOG_FILENAME}</File>
10      <encoder>
11        <pattern>%-14p %d{yyyy/MM/dd HH:mm:ss} [%t] (%F:%L\\) [%M(\\)] - %m%n</pattern>
12      </encoder>
13      <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
14        <maxIndex>5</maxIndex>
15        <FileNamePattern>${LOG_FILENAME}.%i</FileNamePattern>
16      </rollingPolicy>
17      <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
18        <MaxFileSize>20MB</MaxFileSize>
19      </triggeringPolicy>
20   </appender>
21   <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
22      <encoder>
23        <pattern>%-14p %d{yyyy/MM/dd HH:mm:ss} [%t] (%F:%L\\) [%M(\\)] - %m%n</pattern>
24      </encoder>
25   </appender>
26   <logger name="org.duracloud" level="DEBUG" additivity="false">
27      <appender-ref ref="DURACLOUD"/>
28   </logger>
29    <root level="WARN">
30      <appender-ref ref="STDOUT"/>
31   </root>
32</configuration>
```

- Notes on the above logback.xml file
    - on line 4, the attribute "debug" applies to displaying configuration information when LogBack starts up if set to "true"
    - on line 4, the attribute "scan" configures LogBack to re-read the given logback.xml every 60 seconds (by default) for updates
    - on line 26, the attribute "additivity" configures the given logger to inherit the configuration of the parent logger, in this case, the root logger
    - on line 26, if the "additivity" attribute were set to "true", all "DURACLOUD" log output would also log to "STDOUT"

DuraCloud REST API

DuraCloud REST API methods:

- Notes
- All Applications
    - Initialize Security Users
- DuraStore
    - Initialize Stores
    - Get Stores
    - Get Spaces
    - Get Space
    - Get Space Metadata
    - Create Space
    - Set Space Metadata
    - Delete Space
    - Get Content
    - Get Content Metadata
    - Store Content
    - Set Content Metadata
    - Delete Content
    - Get Tasks
    - Perform Task
        - Tasks
- DuraService
    - Initialize Services
    - Get Services

Notes

ℹ️ Each of the methods below has specific security requirements. See DuraCloud Security for more information

⛔ Due to an issue which does not properly handle requests redirected from http to https, it is recommended that all REST API requests use https directly.

✅ Examples calling the API defined below with the Unix utility "curl" can be found here

All Applications

<div align="center">**Security Initialization REST Methods**</div>

Initialize Security Users

- Purpose: Allows the initialization of authorized users
- Request:
  - POST https://host:port/durastore/security
  - POST https://host:port/duraservice/security
  - POST https://host:port/duradmin/security
  - POST https://host:port/durareport/security
- Request Body: XML similar to:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<dur:security-users schemaVersion="0.2" xmlns:dur="duracloud.org">
  <security-user>
    <username>username-0</username>
    <password>password-0</password>
    <enabled>true</enabled>
    <accountNonExpired>true</accountNonExpired>
    <credentialsNonExpired>true</credentialsNonExpired>
    <accountNonLocked>true</accountNonLocked>
    <grantedAuthorities>ROLE_USER</grantedAuthorities>
  </security-user>
  <security-user>
    <username>username-1</username>
    <password>password-1</password>
    <enabled>false</enabled>
    <accountNonExpired>false</accountNonExpired>
    <credentialsNonExpired>false</credentialsNonExpired>
    <accountNonLocked>false</accountNonLocked>
    <grantedAuthorities>ROLE_USER ROLE_ADMIN</grantedAuthorities>
  </security-user>
</dur:security-users>
```

- Response Code: 200 (on success)
- Response Body: "Initialization Successful" (on success)

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.

<div align="center">**Initialization REST Methods**</div>

Initialize Stores

- Purpose: Allows the initialization of storage provider accounts
- Request: POST https://host:port/durastore/stores
- Request Body: XML similar to:

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

- Response Code: 200 (on success)
- Response Body: "Initialization Successful" (on success)

Get Stores

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

```
<storageProviderAccounts>
  <storageAcct isPrimary='true'>
    <id>1</id>
    <storageProviderType>AMAZON_S3</storageProviderType>
  </storageAcct>
  <storageAcct isPrimary="false">
    <id>2</id>
    <storageProviderType>RACKSPACE</storageProviderType>
  </storageAcct>
</storageProviderAccounts>
```

Get Spaces

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

```
<spaces>
  <space id="space1" />
  <space id="space2" />
</spaces>
```

Get Space

- Purpose: Provides a listing of the contents of a space along with space metadata
- Request: GET https://host:port/durastore/spaceID ? (storeID) (prefix) (maxResults) (marker)
    - storeID (optional) - ID of the content storage provider to query (default is primary store)
    - prefix (optional) - Only retrieve content ids with this prefix (default is all content ids)
    - maxResults (optional) - The maximum number of content IDs to return in the list (default is 1000)
      note: the maximum allowable value for maxResults is 1000. Any larger value will be reduced to 1000.
    - marker (optional) - The content ID marking the last item in the previous set (default is the first set of ids)
- Response Code: 200 (on success)
- Response Body: XML similar to:

```
<space id="space1">
  <item>Image 1</item>
  <item>Image 2</item>
</space>
```

- Response Headers: All available space metadata, example:

```
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 https://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 https://host:port/durastore/spaceID ? (storeID)
- Request Headers: Metadata about the space, example:

```
x-dura-meta-space-access: OPEN
   x-dura-meta-custom-metadata: Custom Metadata Value
```

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

```
Location: https://myhost:8080/durastore/space1
```

Set Space Metadata

- Purpose: Updates the metadata associated with a space
- Request: POST https://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 https://host:port/durastore/spaceID ? (storeID)
- Response Code: 200 (on success)
- Response Body: "Space $spaceID deleted successfully" (on success)

Content REST Methods

Get Content

- Purpose: Retrieves a piece of content along with its metadata
- Request: GET https://host:port/durastore/spaceID/contentID ? (storeID) (attachment)
    - if attachment param value is true, a Content-Disposition header is included with the response
- 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: 345670923478509747383839202
  ETag: 345670923478509747383839202
  x-dura-meta-content-name: Testing Content
  x-dura-meta-content-owner: JSmith
```

Get Content Metadata

- Purpose: Retrieves the metadata of a piece of content without the content itself
- Request: HEAD https://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 https://host:port/durastore/spaceID/contentID ? (storeID)
- Request Body: Content to be added
- Request Headers: Metadata about the content, example:

```
Content-Type: text/plain
  Content-MD5: 4cd56e137a93a1accb43c5d32f4afffb
  x-dura-meta-content-name: Testing Content
  x-dura-meta-content-owner: JSmith
```

- Note that when the optional Content-MD5 header is included, the final checksum of the stored file is compared against the MD5 value included in the header to ensure that the file was stored correctly. If the header is not included, an MD5 checksum is computed as the file is transferred to storage, and that value is used in the final comparison.
- Response Code: 201 (on success)
- Response Headers: Location of the new content (i.e. the URL used to create the content), example:

```
Location: https://myhost:8080/durastore/space1/content1
```

Set Content Metadata

- Purpose: Updates the metadata associated with a piece of content
- Request: POST https://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 https://host:port/durastore/spaceID/contentID ? (storeID)
- Response Code: 200 (on success)
- Response Body: "Content $contentID deleted successfully"

Task REST Methods

ℹ️ Tasks are used to perform storage provider actions which cannot be performed in a generic manner across multiple providers.

Get Tasks

- Purpose: Provides a listing of all of the supported tasks for a given provider. Note that if no storeID parameter is included, the task listing is provided for the primary storage provider.
- Request: GET https://host:port/durastore/task ? (storeID)
- Response Code: 200 (on success)
- Response Body: XML similar to:

```
<list>
  <string>task1</string>
  <string>task2</string>
</list>
```

Perform Task

- Purpose: Performs a particular task. Note that most tasks can be performed by only one storage provider type.
- Request: POST https://host:port/durastore/task/taskName ? (storeID)
- Request Body: Parameters for task. Each task will expect parameters in a specific format, see task listing for more details.
- Response Code: 200 (on success)
- Response Body: Response value for task, format varies by task.

Tasks

taskName	Storage Provider	Name	Description	Request Body	Response Body
enable-streaming	Amazon S3	Enable Streaming task	Enables RTMP streaming for all files within a DuraCloud space through the use of Amazon's Cloudfront streaming capability. This task may take up to 15 minutes to complete.	Name of the space for which streaming is to be enabled	Text indicating the results of the task, including the streaming host
disable-streaming	Amazon S3	Disable Streaming task	Disables streaming by removing the ability for Cloudfront to access files within a space. This does not remove the streaming distribution, only disables its use, so enabling streaming on the same space again can be performed much more quickly. Some content in the space may continue to be available for streaming up to 24 hours after streaming has been disabled.	Name of the space for which streaming is to be disabled	Text indicating the results of the task
delete-streaming	Amazon S3	Delete Streaming task	Removes a streaming distribution created by the enable-streaming task. This task should be performed after performing the disable-streaming task. This task may take up to 15 minutes to complete, after which no content in the space will be available for streaming.	Name of the space for which streaming is to be deleted	Text indicating the results of the task
run-hadoop-job	Amazon S3	Run Hadoop Job task	Runs a hadoop job using Amazon's Elastic Map Reduce feature. A JAR which implements the hadoop interfaces is expected to have already been loaded into S3. This JAR is used to execute the hadoop job.	A map serialized into XML which includes, at a minimum, values for jarContentId, sourceSpaceId, destSpaceId, and workSpaceId.	A map serialized into XML which includes the jobFlowId
describe-hadoop-job	Amazon S3	Describe Hadoop Job task	Retrieves information about a hadoop job running in Amazon's Elastic Map Reduce	The Job Flow ID	A map serialized into XML which includes information about the running job
stop-hadoop-job	Amazon S3	Stop Hadoop Job task	Stops a hadoop job running in Amazon's Elastic Map Reduce	The Job Flow ID	A map serialized into XML which includes a results key with a value of either success or failure
noop	Amazon S3	Test task	Provides a simple way to test the calling of tasks	Body content is ignored	Text indicating successful task completion

DuraService

Purpose: DuraService is the application through which DuraCloud manages services. The DuraService REST API provides the means by which services available in the DuraCloud service repository are deployed, configured, and undeployed.

Resources: XML schema which define the service configuration can be found on the Downloads page

Initialization REST Methods

Initialize Services

- Purpose: Initializes the DuraService application
- Request: POST https://host:port/duraservice/services
- Request Body: XML similar to:

```
<servicesConfig>
    <primaryServiceInstance>
        <host>[PRIMARY-SERVICE-INSTANCE-HOST]</host>
        <servicesAdminPort>[PRIMARY-SERVICES-ADMIN-PORT]</servicesAdminPort>
        <servicesAdminContext>[PRIMARY-SERVICES-ADMIN-CONTEXT]</servicesAdminContext>
    </primaryServiceInstance>
    <userStorage>
      <host>[USER-STORAGE-HOST-NAME]</host>
      <port>[USER-STORAGE-PORT]</port>
      <context>[USER-STORAGE-CONTEXT]</context>
      <msgBrokerUrl>[USER-STORAGE-MSG-BROKER-URL]</msgBrokerUrl>
    </userStorage>
    <serviceStorage>
      <host>[SERVICES-STORAGE-HOST-NAME]</host>
      <port>[SERVICES-STORAGE-PORT]</port>
      <context>[SERVICES-STORAGE-CONTEXT]</context>
      <spaceId>[SERVICES-STORAGE-SPACE-ID]</spaceId>
    </serviceStorage>
    <serviceCompute>
      <type>AMAZON_EC2</type>
      <imageId>[MACHINE-IMAGE-ID]</imageId>
      <computeProviderCredential>
        <username>[USERNAME]</username>
        <password>[PASSWORD]</password>
      </computeProviderCredential>
    </serviceCompute>
  </servicesConfig>
```

- Response Code: 200 (on success)
- Response Body: "Initialization Successful" (on success)

Service REST Methods

Get Services

- Purpose: Retrieves a listing of services, along with their configuration options
- Request: GET https://host:port/duraservice/services ? (show)
    - Parameter options for show (optional)
        1. available (default) - Includes only services which have not been deployed but are available for deployment
        2. deployed - Includes only services which have been deployed and started
- Response Code: 200 (on success)
- Response Body: XML list of services (see service config xsd)

Get Service

- Purpose: Retrieves information about a particular service including description, configuration options, and all deployments
- Request: GET https://host:port/duraservice/serviceID
- Response Code: 200 (on success)
- Response Body: XML service (see service config xsd)

Get Deployed Service

- Purpose: Retrieves information about a deployed service including description, configuration options, and a single deployment indicating the configuration options in use
- Request: GET https://host:port/duraservice/serviceID/deploymentID
- Response Code: 200 (on success)
- Response Body: XML service (see service config xsd)

Get Deployed Service Properties

- Purpose: Retrieves the runtime properties of a deployed service
- Request: GET https://host:port/duraservice/serviceID/deploymentID/properties
- Response Code: 200 (on success)
- Response Body: XML service (simple xml Map serialization)

Deploy Service

- Purpose: Deploys and starts an available service
- Request: PUT https://host:port/duraservice/serviceID ? (serviceHost)
    - Parameter value for serviceHost (optional) should indicate the services host on which the service should be deployed. Default is the primary customer host.
- Request Body: XML user configuration indicating the config selections for the service (see user config portion of service config xsd)
- Response Code: 201 (on success)
- Response Header: Location header indicates the URL at which information about the deployed service can be retrieved (the URL for a get deployed service call) which includes the deploymentID

Update Service Configuration

- Purpose: Updates the configuration of a deployed service
- Request: POST https://host:port/duraservice/serviceID/deploymentID
- Request Body: Updated XML user configuration indicating the config selections for the service (see user config portion of service config xsd)
- Response Code: 200 (on success)

UnDeploy Service

- Purpose: Stops and Undeploys a deployed service
- Request: DELETE https://host:port/duraservice/serviceID/deploymentID
- Response Code: 200 (on success)

DurAdmin

Purpose: DurAdmin is the user-facing application through which DuraCloud exposes DuraStore and DuraService functionality. The DurAdmin REST API provides the means by which DurAdmin is initialized.

Initialization REST Methods

Initialize Application

- Purpose: Allows the initialization of duradmin
- Request: POST https://host:port/duradmin/init
- Request Body: XML similar to:

```
<duradminConfig>
  <durastoreHost>[host]</durastoreHost>
  <durastorePort>[port]</durastorePort>
  <durastoreContext>durastore</durastoreContext>
  <duraserviceHost>[host]</duraserviceHost>
  <duraservicePort>[port]</duraservicePort>
  <duraserviceContext>duraservice</duraserviceContext>
</duradminConfig>
```

- Response Code: 200 (on success)
- Response Body: "Initialization Successful" (on success)

DuraReport

Purpose: DuraReport generates reports relating to the status of your DuraCloud instance, and provides a simple interface for accessing those reports.

Resources: XML schema which defines the expected transfer data for storage and service reporting can be found on the Downloads page

Initialization REST Methods

Initialize Application

- Purpose: Allows the initialization of durareport
- Request: POST https://host:port/durareport/reports
- Request Body: XML similar to:

```
<durareportConfig>
  <durastoreHost>[host]</durastoreHost>
  <durastorePort>[port]</durastorePort>
  <durastoreContext>durastore</durastoreContext>
  <duraserviceHost>[host]</duraserviceHost>
  <duraservicePort>[port]</duraservicePort>
  <duraserviceContext>duraservice</duraserviceContext>
</durareportConfig>
```

- Response Code: 200 (on success)
- Response Body: "Initialization Successful" (on success)

Storage Report REST Methods

Get Latest Storage Report

- Purpose: Provides the most current storage report in XML format
- Request: GET https://host:port/durareport/storagereport
- Response Code: 200 (on success)
- Response Body: XML, defined by the storage report XSD

Get Storage Report List

- Purpose: Provides a list of all storage report IDs
- Request: GET https://host:port/durareport/storagereport/list
- Response Code: 200 (on success)
- Response Body: XML, defined by the storage report XSD

Get Storage Report

- Purpose: Provides a specific storage report based on the provided report ID
- Request: GET https://host:port/durareport/storagereport/reportID
- Response Code: 200 (on success)
- Response Body: XML, defined by the storage report XSD

Get Storage Report Info

- Purpose: Provides a information about the current status of the storage reporting system
- Request: GET https://host:port/durareport/storagereport/info
- Response Code: 200 (on success)
- Response Body: XML, defined by the storage report XSD

Start Storage Report

- Purpose: Starts a storage report if one is not already running
- Request: POST https://host:port/durareport/storagereport
- Response Code: 200 (on success)
- Response Body: "Report Started" (on success), or ""Report Already In Progress" (if a report is already in progress)

Cancel Storage Report

- Purpose: Cancels a running storage report
- Request: DELETE https://host:port/durareport/storagereport
- Response Code: 200 (on success)
- Response Body: "Storage report cancelled"

Schedule Storage Report

- Purpose: Schedules a time for a storage report to be run
- Request: POST https://host:port/durareport/storagereport/schedule ? (startTime) (frequency)
    - startTime: time (in milliseconds since the epoch) to begin the next storage report
    - frequency: time (in milliseconds) to wait between running reports (minimum value is 600000)
- Response Code: 200 (on success)
- Response Body: "Storage reports scheduled" (on success)

Cancel Storage Report Schedule

- Purpose: Cancels all entries on the storage report schedule
- Request: DELETE https://host:port/durareport/storagereport/schedule
- Response Code: 200 (on success)
- Response Body: "Storage Reports schedule cancelled"

Service Report REST Methods

Get Deployed Services Report

- Purpose: Provides a listing of the services which are currently deployed
- Request: GET https://host:port/durareport/storagereport/deployed
- Response Code: 200 (on success)
- Response Body: XML, defined by the service report XSDs

Get Completed Services Report

- Purpose: Provides a listing of the most recent completed services
- Request: GET https://host:port/durareport/servicereport ? (limit)
  - Parameter value for limit (optional) should indicate the maximum number of services to return in the report (default is 20, max is 1000)
- Response Code: 200 (on success)
- Response Body: XML, defined by the service report XSDs

Get Completed Services Report List

- Purpose: Provides a listing of all service report IDs
- Request: GET https://host:port/durareport/servicereport/list
- Response Code: 200 (on success)
- Response Body: XML, defined by the service report XSDs

Get Services Report

- Purpose: Provides a specific services report based on the provided report ID
- Request: GET https://host:port/durareport/servicereport/reportID
- Response Code: 200 (on success)
- Response Body: XML, defined by the service report XSDs

REST API Examples Using curl

Convenience Variables

The curl commands below can be used directly if you define the following variables in your shell

```
host=<duracloud-hostname>
space-0=<any-name>
space-1=<any-name>
user=<username>
pword=<password>
file=<any-file-name>
```

DuraStore Notes

The curl commands in the DuraStore section expect a test file for uploads.

```
echo hello > ${file}
```

Note that if the target of a content or space retrieval (GET) has access permissions set to "OPEN", then the "-u" option in the curl commands is not required.

DuraStore

Get Stores

```
curl -u ${user}:${pword} https://${host}/durastore/stores
```

Get Spaces

```
curl -u ${user}:${pword} https://${host}/durastore/spaces
curl -u ${user}:${pword} https://${host}/durastore/spaces?storeID=1
```

Create Space

```
curl -u ${user}:${pword} -X PUT https://${host}/durastore/${space-0}
curl -u ${user}:${pword} -H "x-dura-meta-city: arlington" -H "x-dura-meta-state: va" -X PUT
https://${host}/durastore/${space-1}?storeID=1
```

Store Content

```
curl -u ${user}:${pword} -T ${file} https://${host}/durastore/${space-0}/test.txt
curl -u ${user}:${pword} -T ${file} https://${host}/durastore/${space-0}/item.txt
```

Get Space

```
curl -u ${user}:${pword} https://${host}/durastore/${space-0}
curl -u ${user}:${pword} https://${host}/durastore/${space-1}?storeID=1
curl -u ${user}:${pword} https://${host}/durastore/${space-0}?prefix=test
```

Set Space Metadata

```
curl -u ${user}:${pword} -H "x-dura-meta-country: usa" -X POST https://${host}/durastore/${space-0}
```

Get Space Metadata

```
curl -u ${user}:${pword} -I https://${host}/durastore/${space-0}
curl -u ${user}:${pword} -I https://${host}/durastore/${space-1}?storeID=1
```

Get Content

```
curl -u ${user}:${pword} https://${host}/durastore/${space-0}/test.txt
curl -u ${user}:${pword} https://${host}/durastore/${space-0}/test.txt?storeID=0\&attachment=true
```

Set Content Metadata

```
curl -u ${user}:${pword} -X POST -H "x-dura-meta-color: green"
https://${host}/durastore/${space-0}/test.txt
```

Get Content Metadata

```
curl -u ${user}:${pword} -I https://${host}/durastore/${space-0}/test.txt
```

Delete Content

```
curl -u ${user}:${pword} -X DELETE https://${host}/durastore/${space-0}/test.txt
```

Delete Space

```
curl -u ${user}:${pword} -X DELETE https://${host}/durastore/${space-0}
curl -u ${user}:${pword} -X DELETE https://${host}/durastore/${space-1}?storeID=1
```

DuraService Notes

The commands in the DuraService section below demonstrate the deployment, inspection, reconfiguration, and undeployment of a service.
As of release 0.8.0, the Bit Integrity Checker has service id of '0' (noted in variable below).
The actual deployment id is dynamically generated based on the number of previous deployments within a given DuraCloud application.
After deploying the service, use the "Get Deployed Service" call to determine the specific *deployment id* of your deployed service.

```
bitintegrity=0
deployment=<determined-by-inspection>
```

The following two service configuration files are provided for deployment and reconfiguration of the Bit Integrity Checker.
They are based on the spaces that were created in the DuraStore section, but can be modified to indicate other configuration options or to execute over alternate spaces.

1. configuration file 0
2. configuration file 1

DuraService

Get Services

```
curl -u ${user}:${pword} https://${host}/duraservice/services
curl -u ${user}:${pword} https://${host}/duraservice/services?show=deployed
```

Deploy Service

```
curl -u ${user}:${pword} -X PUT -T deploy.xml https://${host}/duraservice/${bitintegrity}
```

Get Service

```
curl -u ${user}:${pword} https://${host}/duraservice/${bitintegrity}
```

Get Deployed Service

```
curl -u ${user}:${pword} https://${host}/duraservice/${bitintegrity}/${deployment}
```

Get Deployed Service Properties

```
curl -u ${user}:${pword} https://${host}/duraservice/${bitintegrity}/${deployment}/properties
```

Update Service Configuration

```
curl -u ${user}:${pword} -X PUT -T deploy-new.xml https://${host}/duraservice/${bitintegrity}
```

UnDeploy Service

```
curl -u ${user}:${pword} -X DELETE https://${host}/duraservice/${bitintegrity}/${deployment}
```

Release Notes

Release 1.0

Released: July 29, 2011

The primary features of release 1.0 are:

- Storage Reporting
    - A new feature which provides detailed information about the amount of data you have stored in DuraCloud, as well as the kinds of data you have stored in DuraCloud. Reports are generated automatically and the information can be viewed using the DuraCloud dashboard or downloaded for processing using new REST API methods.
- Service Reporting
    - A new feature which provides detailed information about services which are currently running and which have run previously in your DuraCloud account. Reports are generated automatically as services are run and the information can be viewed using the DuraCloud dashboard or downloaded for processing using new REST API methods.
- Service Dependencies
    - Any service can now define a dependency on another service.
    - "System" services, which need to be installed prior to other services being deployed, are now installed on-the-fly only when needed.
- Improved service feedback
    - The information provided by running services is now more complete and more consistent with other DuraCloud services.
- Improved character set support
    - Content IDs can now consist of any characters which can be properly encoded using UTF-8 (with the exception of "reserved" characters mentioned here)
- Image Viewer URL stability
    - The URLs for images made available by the Image Server service will now stay consistent across restarts of the service and restarts of the DuraCloud instance.

For more details about specific changes in release 1.0, see the JIRA issue tracker.

Release 0.9

Released: April 27, 2011

The primary features of release 0.9 are:

- The Duplicate on Ingest service is now the Duplicate on Change service.
    - This service now supports all of the same on-ingest features as before, but it now also performs duplication of all update and delete actions as well. This allows the primary and secondary cloud stores to be kept completely in sync.
- The Bulk Bit Integrity Service has been improved.
    - This service has been updated and verified to properly handle spaces with up to 1 million items
    - The second step of the MD5 verification, which used to run locally on the instance, has been moved to hadoop, allowing the service to complete much more quickly for large data sets.
- User management functions have been removed, as they are now performed by the DuraCloud Management Console.

- - As a convenience, administrators are still able to see the list of users and their roles within the DuraCloud Administrator UI.
  - Service outputs have been made more consistent.
    - All DuraCloud services which produce an output file now store that file in the x-service-out space.
    - Services which produce log files store those logs in the x-service-work space.
    - The names of the output files have been made more consistent, making it simpler to determine which files correspond to which service deployment.
  - Password security has been improved.
    - All passwords used within DuraCloud are now immediately pushed through a hashing function before being are stored, so that no user passwords are transferred or stored as clear text.
  - A ServiceClient is now available, to compliment the StoreClient and make it easier to make direct API calls to manipulate DuraCloud services.

For more details about specific changes in release 0.9, see the JIRA issue tracker.

Release 0.8

Released: Jan 26, 2011

The primary features of release 0.8 are:

- Simplified services
  - The listing of services has been better organized, to make finding the service you would like to run simpler.
  - All services now require you to set fewer options, simplifying the deployment process.
  - Bulk services (Image Transformer - Bulk, Bit Integrity Checker - Bulk, and Duplicate on Demand) now provide a *standard* configuration mode which handles the setting of server type, and number of servers used to perform the job, so that you no longer have to make those choices.
  - The output location for services has been set to the *x-service-out* space, which removes the need to set this value for each service, and provides a standard location to look for service output reports.
  - The work location for services has been set to the *x-service-work* space, which removes the need to set this value for each service, and provides a standard location to look for service logs and other run time artifacts.
- More reliable services
  - Several bugs which have caused services to fail have been resolved.
- Sync Tool command line flags now match those offered by the Retrieval Tool.
- UI updates which provide better visual cues for which storage provider is in use.
- A host of bug fixes and small tweaks

For more details about specific changes in release 0.8, see the JIRA issue tracker.

Release 0.7

Released: Oct 28, 2010

The primary features of release 0.7 are:

- A new Retrieval Tool, a companion to the existing Sync Tool, which is a command-line tool for the retrieving content from DuraCloud spaces.
- A new Bulk Bit Integrity Checker service, which can be run over content stored in Amazon to create a listing of checksums calculated for each file. This new service pairs well with the Bit Integrity Checker service (previously known as the Fixity Service), allowing the heavy processing to be handled in parallel using Hadoop on an Amazon EC2 cluster, and the simpler checks and comparisons to be handled by the DuraCloud instance.
- A new Duplicate on Demand service, which can be used to copy files from the primary Amazon store into another storage provider. This service pairs well with the Duplicate on Upload service (previously known as the Replication Service) by performing the large up-front copy using Hadoop on an Amazon EC2 cluster, then allowing Duplicate on Upload to watch for and add new files as they are uploaded.
- Integration of a new storage provider: Microsoft Windows Azure.

For more details about specific changes in release 0.7, see the JIRA issue tracker.

Note that there have been issues discovered during testing of the Bulk Image Transformer (included in release 0.6 as the Bulk Image Conversion Service). If you choose to run this service, it is recommended that the size of images being used be kept under 100MB. The likelihood of success appears to increase with server size, and number of servers being set to 3 or more is recommended. If you do run this service, please note the data set and configuration and make us aware of the outcome.

Release 0.6

Released: Sept 03, 2010

The primary features of release 0.6 are:

- Addition of a new Fixity Service, which allows for bit integrity checking on content stored within DuraCloud. This service has many options

to fit various usage needs. For more information, see the Fixity Service page.
- Addition of a new Bulk Image Conversion Service, which, like the Image Conversion Service, allows for converting images into other formats. This new service, however, makes use of Hadoop in the background to run the conversion using multiple servers, allowing for much higher overall throughput.
- An updated handling of space metadata so that spaces with a large number of content items will not cause slow response times. Now spaces with more than 1000 items will initially show a value of 1000+ as the number of items in the space. DurAdmin, the administrative interface, will then calculate the total number of items on the fly.
- The Sync Tool has a new option (-e) which will cause the tool to exit once it has completed syncing rather than continually monitoring for changes. This makes it easier for administrators to include the Sync Tool in scripts which run daily or weekly to ensure all local content is moved to DuraCloud.
- DurAdmin now provides a way to delete groups of content items and spaces in one step.
- A host of bug fixes and small tweaks

For more details about specific changes in release 0.6, see the JIRA issue tracker.

Release 0.5

Released: July 28, 2010

The primary feature of release 0.5 is the addition of a completely new administrative user interface. This UI, called DurAdmin like its predecessor, provides for easy access to the primary features of DuraCloud.

For more details about specific changes in release 0.5, see the JIRA issue tracker.

Release 0.4

Released 0.4.1: June 30, 2010

- This build release is the first publicly available
- It primarily provides clean-up of projects and tests

Released 0.4: June 21, 2010

The primary features added in release 0.4 of DuraCloud were:

- Media Streaming Service
  - Provides a way to enable streaming for video and audio files as well as providing an example media player.
- Logging moved to SLF4J over Logback
  - Provides greater consistency in log output and greater flexibility in log configuration

For more details about specific changes in release 0.4, see the JIRA issue tracker.

Release 0.3

Released: May 17, 2010

The primary features in the third pilot release of DuraCloud are:

- Security
  - All DuraCloud applications now require authentication prior to performing write activities
  - Read activities on 'closed' spaces also require authentication, but 'open' spaces allow anonymous read access
- Sync Tool
  - Provides a command line utility for keeping DuraCloud content synchronized with the local file system

Other improvements in the 0.3 release:

- Image Conversion Service
  - Adds an option to convert images to the (web standard) sRGB color space
  - Adds the capability to perform multiple conversions at once (providing the compute capacity is available) and provides more frequent activity feedback through the continual writing of the conversion output file
- DuraStore
  - Adds an option for users to provide MD5 checksum when adding content. This disables the in-transfer MD5 computation (providing improved performance) and compares the final MD5 computed by the storage provider with the user provided MD5.

For more details about specific changes in release 0.3, see the JIRA issue tracker.

Release 0.2

Released: Feb 19, 2010

The second pilot release of DuraCloud focused on providing access to services which can be run over content, as well as improvements to the storage foundation provided by the first release.

Services available as of release 0.2:

- J2K service - serves J2K images, provides a J2K image viewer
- Image Conversion service - converts image files from one format to another
- Replication service - replicates content stored in one provider to another upon content upload
- Web Application Utility service - infrastructure service required by J2K service (allows for deployment of web applications)
- ImageMagick service - infrastructure service required by Image Conversion service (provides access to ImageMagick utilities)

Service functions available as of release 0.2:

- Services may be deployed with configuration
- Available and deployed services may be listed
- Deployed service configuration may be viewed and updated
- Deployed service properties may be viewed
- Deployed services may be undeployed and redeployed

New storage functions available as of release 0.2:

- Space content may be listed in chunks with an optional prefix filter
- Space and content metadata may be edited via the UI
- Space and content metadata tags may be added/removed via the UI

For more details about specific changes in release 0.2, see the JIRA issue tracker. Note that while most items included in the release are listed in the tracker, we migrated to using JIRA while working on release 0.2, so issues completed prior to the migration are not included.

Release 0.1

Released: Nov 2, 2009

The first pilot release of DuraCloud laid 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*

DuraCloud Sync Tool

Introduction

The Sync Tool is a utility which was created in order to provide a simple way to move files from a local file system to DuraCloud and subsequently keep the files in DuraCloud synchronized with those on the local system.

Download

Download the sync tool here from the Downloads page.

How the Sync Tool Works

- When you run the Sync Tool for the first time, you must include DuraCloud connection information (host, port, username, password) as well as the space where you would like all of your files stored. You must also provide a list of directories which will be synced to DuraCloud and a directory for the Sync Tool to use for its own work.
- When the Sync Tool starts up, it will look through all of the files in each of the local content directories and add them to its internal queue for processing. Each of those files will then be written to your DuraCloud space. As this initial write is happening a listener is set up to watch for any file changes within each of the content directories. As a change occurs (a file is added, updated, or deleted), that change is added to the queue, and the appropriate action is taken to make the DuraCloud space consistent with the local file (i.e. the file is either written to the space or deleted from the space.)
- You can stop the Sync Tool at any time by typing 'x' or 'exit' on the command line where it is running. It will stop all listeners, complete

any file transfers that are in progress, and close down.
- When you restart the Sync Tool, if you point it at the same work directory, it will pick up where it left off. While the Sync Tool is running, it is constantly writing backups of its internal queue, so it first reads the most current backup and begins processing the files there. It then scans the content directories to see if there are any files which have been added or updated since the last backup, and it also pulls a list of files from the DuraCloud space and scans that list to see if any local files have been deleted. Any changes detected are added to the internal queue, and the Sync Tool continues to run as usual.

Operational notes

- Restarting
  - You can perform a restart of the Sync Tool by using the -g command line option to point to the Sync Tool configuration file, which is written into the work directory (named synctool.config)
  - If you would like the Sync Tool to perform a clean start rather than a restart (i.e. you would like it to compare all files in the content directories to DuraCloud) you will need to either point it to a new work directory, or clear out the existing work directory.
  - The Sync Tool will perform a clean start (not a restart) if the list of content directories is not the same as the previous run. This is to ensure that all files in all content directories are processed properly.
- Collisions
  - The Sync Tool allows you to sync multiple local directories into a single space within DuraCloud. Because of this, there is the possibility of file naming collisions, where two local files resolve to the same DuraCloud ID. If this happens, one file will be overwritten by the other. There are a few ways to ensure that this does not occur:
    - Ensure that the top level files and directories within the set of content directories do not have overlapping names.
    - Sync only a single directory to a space. You can run multiple copies of the Sync Tool, each over a single local directory, syncing to its own DuraCloud space.
- Work Directory - these files and directories can be found in the work directory (specified using the -w command line parameter)
  - Config Files
    - When the Sync Tool starts up, it writes the list of parameters and values provided by the user on startup to a file called synctool.config in the work directory. This file can be used to restart the Sync Tool, using the -g parameter to point to the file's location. You can also restart the Sync Tool by indicating the same set of options as used originally. The -g parameter is for convenience only and is not required in any circumstance. Note that this file is overwritten each time the Sync Tool is run with a different set of parameters, so you may choose to copy the file elsewhere (or give it a new name) if you would like to keep a copy of a particular configuration set.
    - You may also see a file named synctool.config.bak in the work directory which is used to compare against the current config in order to determine if a restart is possible. In order for a restart to occur, the list of content directories (-c parameter) must be the same as the previous execution of the tool, and there must be at least one changed list backup (see below.)
  - Changed List Directory
    - While the Sync Tool is running it is constantly updating the list of files which have been changed (when starting the first time, this includes all files in the directories that need to be synced). In order to allow the Sync Tool to restart after it has been stopped, this list of files is continually backed up into the *changedList* directory. There is no reason to edit these files, but you may choose to delete the *changedList* directory along with the config files mentioned above to ensure that the Sync Tool does not attempt to perform a restart.
  - Logs Directory
    - Information about what the Sync Tool is doing while it is running can be found in the sync-tool.log file. It is a good idea to monitor this file for errors and warnings as this information is not printed to the console.
    - The duracloud.log file is useful for application debugging when the information in the sync-tool.log file is insufficient to understand a problem.

Prerequisites

- You must have Java version 6 or above installed on your local system. If Java is not installed, you will need to download and install it. To determine if the correct version of Java is installed, open a terminal or command prompt and enter

```
java -version
```

The version displayed should be 1.6.0 or above. If running this command generates an error, Java is likely not installed.
- You must have downloaded the Sync Tool. It is available as a link near the top of this page.

Starting the Sync Tool

- To run the Sync Tool, open a terminal or command prompt and navigate to the directory where the Sync Tool is located
- To display the help for the Sync Tool, run

```
java -jar synctool-1.0.0-driver.jar
```

- When running the Sync Tool for the first time, you will need to use these options:

Short Option	Long Option	Argument Expected	Required	Description	Default Value (if optional)
-h	--host	Yes	Yes	The host address of the DuraCloud DuraStore application	
-r	--port	Yes	No	The port of the DuraCloud DuraStore application	443
-i	--store-id	Yes	No	The Store ID for the DuraCloud storage provider	The primary storage provider is used
-s	--space-id	Yes	Yes	The ID of the DuraCloud space where content will be stored	
-u	--username	Yes	Yes	The username necessary to perform writes to DuraStore	
-p	--password	Yes	Yes	The password necessary to perform writes to DuraStore	
-c	--content-dirs	Yes	Yes	A list of the directory paths to monitor and sync with DuraCloud. If multiple directories are included in this list, they should be separated by a space.	
-w	--work-dir	Yes	Yes	The state of the sync tool is persisted to this directory	
-f	--poll-frequency	Yes	No	The time (in ms) to wait between each poll of the sync-dirs	10000 (10 seconds)
-t	--threads	Yes	No	The number of threads in the pool used to manage file transfers	3
-m	--max-file-size	Yes	No	The maximum size of a stored file in GB (value must be between 1 and 5), larger files will be split into pieces	1
-d	--sync-deletes	No	No	Indicates that deletes performed on files within the content directories should also be performed on those files in DuraCloud; if this option is not included all deletes are ignored	Not set
-x	--exit-on-completion	No	No	Indicates that the sync tool should exit once it has completed a scan of the content directories and synced all files; if this option is included, the sync tool will not continue to monitor the content dirs	Not set

- When the Sync Tool runs, it creates a backup of your configuration in the work directory that you specify. When running the tool again, you can make use of this file to keep from having to re-enter all of the options specified on the initial run. In this case you need only a single option:

Short Option	Long Option	Argument Expected	Required	Description
-g	--config-file	Yes	Yes	Read configuration from this file (a file containing the most recently used configuration can be found in the work-dir, named synctool.config)

- An example for running the Sync Tool

```
java -jar synctool-1.0.0-driver.jar -w C:\tools\synctool\backup -c C:\files\important -f 2000 -h
test.duracloud.org -s important-dir-backup -t 5 -u myname -w mypassword
```

Runtime commands

- While the Sync Tool is running, these commands are available. Just type them on the command line where the tool is running.

Short Command	Long Command	Description
x	exit	Tells the Sync Tool to end its activity and close
c	config	Prints the configuration of the Sync Tool (the same information is printed at startup)
s	status	Prints the current status of the Sync Tool
l <Level>	N/A	Changes the log level to <Level> (may be any of DEBUG, INFO, WARN, ERROR)

h	help	Prints the runtime command help

DuraCloud Java Clients

Download

Download the Store Client, Service Client, and Report Client from the Downloads page

Available clients

StoreClient

DuraCloud provides access to files stored in cloud storage systems through an application called DuraStore. DuraStore is installed and running on your DuraCloud instance and can be accessed via a REST interface. In order to aid Java developers in communicating with DuraStore, a Java client, called StoreClient was written.

ServiceClient

DuraCloud provides management capabilities to deploy, undeploy, configure, and gain status of the services within DuraCloud through an application called DuraService. DuraService is installed and running on your DuraCloud instance and can be accessed via a REST interface. In order to aid Java developers in communicating with DuraService, a Java client, called ServiceClient was written.

ReportClient

DuraCloud provides reporting capabilities which assist in understanding and managing the content stored in DuraCloud and services run through DuraCloud. These capabilities are included in an application called DuraReport. DuraReport is installed and running on your DuraCloud instance and can be accessed via a REST interface. In order to aid Java developers in communicating with DuraReport, a Java client, called ReportClient was written.

Using the clients

To use any of the above clients, you will need all of the jars included in the libs directory of the download package to be available on your classpath. You will then be able to write code using the provided Javadocs to interact with the client.

Example code

For each of the clients listed above, an example Java class has been provided to assist in set up and testing, as well as a starting point for writing your client code. The example client (found in ExampleClient.java) includes a simple main class which performs a subset calls the client is capable of performing, and printing information to the console. To run an example:

- Extract the client zip file
- Update the HOST, PORT, USERNAME, and PASSWORD constant values in ExampleClient.java as needed to connect to your DuraCloud instance.
- Make sure Ant is available on your path and type "ant" to compile the example.
- Type "ant run" to run the example.

DuraCloud Administration

This document details some of the considerations of concern to a DuraCloud administrator.

Naming restrictions

1. Space names
   a. The following restrictions apply to user-defined space names
      - only lowercase letters, numbers, periods, and dashes
      - no adjacent pair of "-" and/or "."
      - no number immediately following the last "."
      - between 3 and 63 characters
      - must start with a letter
      - may not end with a dash
   b. Note: Users can provide space names through the REST-API that do not follow these conventions, but the space actually

created will have a different name under the covers.
2. Reserved space names
   a. Due to some specific operations exposed through the durastore REST-API, the following names are unavailable as user-defined space names
      - stores
      - spaces
      - security
      - task
3. Content object names
   a. The only restrictions are that a content object name
      - cannot include a question mark '?' character
      - cannot include a reverse solidus (backslash) '\' character
      - cannot include a semicolon ';' character
      - is limited to 1024 bytes (byte count is checked after URL and UTF-8 encoding)

DuraCloud Services

- Introduction
- Duplicate on Change
- Duplicate on Demand
- Image Server
- Media Streamer
- Bit Integrity Checker
- Bit Integrity Checker - Tools
- Bit Integrity Checker - Bulk
- Image Transformer
- Image Transformer - Bulk

Introduction

The DuraCloud application provides a set of services which can be deployed and used for a variety of purposes, primarily to process the content which has been loaded into DuraCloud storage. The following list of services describes how each service is expected to be used and the options available for tailoring the service to your needs.

Note that the current way all DuraCloud services are configured, they will not auto-restart if they fail during processing. If you notice a failed job state, simply redeploy the service. Automatic service recovery is on the roadmap for DuraCloud development in the future and will be made available as soon as possible.

Duplicate on Change

Description:

The Duplicate on Change service provides a way to ensure that the content stored in DuraCloud is synchronized between two different storage providers. The Duplicate on Change service duplicates any changes made to spaces, content, or metadata on the store being watched by the service. This means that once the Duplicate on Change service is deployed, it notices all content that is added, updated, or deleted on the watched DuraCloud provider and performs the same functions on a secondary provider. All content that is copied will be placed in an identically named space in the secondary storage location with the same metadata fields attached. The duplication provided by this service is one-way; changes made to the secondary provider will not be reflected in the primary.

Configuration Options:

1. Watch this store: The primary storage location which DuraCloud will monitor for changes. When spaces, content, or metadata are added, updated, or deleted in this store, the same actions will be taken in the secondary store.
2. Apply to this store: The secondary store where changes from the primary store will be applied.

Duplicate on Demand

Description:

The Duplicate On Demand service provides a simple way to duplicate content from one space to another. This service is primarily focused on allowing the duplication of content from the primary storage provider to a secondary provider. To begin, a source space is chosen, along with a store and space to which content will be duplicated. The service then performs a copy of all content and metadata in the source space to the duplication space, creating the space if necessary. When the service has completed its work, a results file will be stored in the chosen space and a set of files (primarily logs) created as part of the process will be stored in the work space.

Configuration Options:

1. Source Space: DuraCloud space where source files can be found
2. Copy to this store: DuraCloud store to which content will be copied

3. Copy to this space: DuraCloud space where content will be copied
4. Standard vs. Advanced configuration
    a. Standard mode automatically sets up the service to be run
    b. Advanced mode allows the user to configure the number and type of servers that will be used to run the job
        i. Number of Server Instances: The number of servers to use to perform the duplication task.
        ii. Type of Server: The type (size) of server used as perform the task. The larger the server, the faster the processing will occur. Larger servers also cost more than smaller servers to run. For more information, see the Amazon EC2 documentation.

Service Output

All outputs of this service are placed in the system space, *x-service-out*.

1. duplicate-on-demand/duplicate-results-<date>.csv
    * Final report indicating which files were duplicated, as well as any failures encountered

Image Server

Description:

The Image Server provides a viewer for image files through use of the Djatoka image server. While this service is geared towards serving JPEG 2000 images, it supports multiple image file types by converting them to JPEG 2000 format on the fly.

Note that the current implementation of this service requires that spaces be set to OPEN in order to use the viewer to view image files.

Configuration Options:

None

Media Streamer

Description:

The Media Streamer provides streaming capabilities for video and audio files. The service takes advantage of Amazon Cloudfront streaming, so files to be streamed must be within a space on an Amazon provider. Also, all media to be streamed by this service needs to be within a single space.

Amazon Cloudfront streaming uses the Flash Media Server to host streaming files over RTMP. **File formats supported include MP3, MP4 and FLV among others.** For a full listing of supported file types see the Flash Media Server documentation.

Configuration Options:

1. Source Media Space: The DuraCloud space where the source video and audio files to be streamed are stored. The Media Streamer service attempts to stream all files in this space.
2. Viewer Space: A DuraCloud space where example viewer files will be stored. After the service has started, this space will include a playlist including all items in the source media space as well as example html and javascript files which can be used to display a viewer.

Output Files, the following files can be found in the configured Viewer Space once the Media Streamer is running:

* player.swf - The flash-based video player JWPlayer
* playlist.xml - A playlist, created by DuraCloud, which includes all of the items in your Source Media Space
* playlistplayer.html - An HTML file, created by DuraCloud, which uses JWPlayer to display the items in the playlist
* singleplayer.html - An HTML file, created by DuraCloud, which uses JWPlayer to display a single media file (typically, the first item in your Source Media Space)
* stylish.swf - A supplementary flash file used to style the JWPlayer
* swfobject.js - A javascript file (available from here) used to embed the JWPlayer on a web page
* viewer.js - A javascript file, created by DuraCloud, used to simplify the loading of JWPlayer

All of the output files are intended as examples only. Their purpose is give developers a starting point for embedding video streamed by DuraCloud on their own web pages. Feel free to use, modify, ignore, or delete these files.

> ℹ The Media Streamer expects that all files to be streamed exist in the Source Media Space when the service is started. If you add files when the service is already running and would like those files to be streamed as well, you will need to restart the Media Streamer.

Bit Integrity Checker

Description:

The Bit Integrity Checker provides the ability to verify that the content held within DuraCloud has maintained its bit integrity. There are two modes

of operation.
Modes:

1. Verify integrity of a Space
2. Verify integrity of an item list

When running in the *Verify integrity of a Space* mode, the checker performs the following steps

- collect the content hash values for each item from the underlying storage provider
- stream through each item recalculating their hashes
- compare the two listings

When running in the *Verify integrity of an item list* mode, the checker performs the following steps

- stream through each item in the provided listing, recalculating their hashes
- compare the newly generated listing with the provided listing

Configuration Options:

1. Stores: The underlying storage provider over which the service will run
2. Space containing content items: The DuraCloud space in which the content items to be verified reside
3. Verify integrity of an item list mode
    a. Input listing name: Name of the content item which contains the listing of items over which to run the service

Service Ouputs
All outputs of this service are placed in the system space, *x-service-out*.

1. bitintegrity/fingerprints-gen-<spaceId>-<date>.csv
    - Interim listing generated with hash values from underlying storage provider
2. bitintegrity/fingerprints-<spaceId>-<date>.csv
    - Interim listing with hashes recalculated from content streams
3. bitintegrity/fixity-report-<spaceId>-<date>.csv
    - Final report with status of integrity check

Bit Integrity Checker - Tools

Description:

The Bit Integrity Checker Tools provide additional bit integrity checking utilities which can be used to perform specific integrity checking tasks.

Modes:

1. Generate integrity information for a Space
2. Generate integrity information for an item list
3. Compare two integrity reports

Configuration Options:

1. Mode 1 - *Generate integrity information for a Space*
    a. Get integrity information from...
        i. The storage provider: Determine the file MD5 by asking the storage provider for its stored MD5 value
        ii. The files themselves: Determine the file MD5 by retrieving them from the storage provider and computing the MD5
    b. Stores: The underlying storage provider in which the following space resides
        i. Space containing content items: The DuraCloud space in which the content items to be considered reside
2. Mode 2 - *Generate integrity information for an item list*
    a. Get integrity information from...
        i. The storage provider: Determine the file MD5 by asking the storage provider for its stored MD5 value
        ii. The files themselves: Determine the file MD5 by retrieving them from the storage provider and computing the MD5
        iii. Input listing name: Name of the content item which contains the listing of items over which to run the service
    b. Stores: The underlying storage provider in which the following space resides
        i. Space with input listing: The DuraCloud space in which the input listing file resides
3. Mode 3 - *Compare two integrity reports*
    a. Input listing name: Name of the first content item which contains a listing of items to be compared to the second listing
    b. Second input listing name: Name of the second content item which contains a listing of items to be compared to the first listing
    c. Stores: The underlying storage provider in which the following spaces reside
        i. Space with input listing: The DuraCloud space in which the first input listing file resides
        ii. Space with second input listing: The DuraCloud space in which the second input listing file resides

Service Ouputs
All outputs of this service are placed in the system space, *x-service-out*.

1. bitintegrity/fingerprints-<spaceId>-<date>.csv
    - Listing of hashes when running in *from space* or *from list* modes
2. bitintegrity/fixity-report-<listingId-0>-vs-<listingId-1>-<date>.csv

- Comparison report of two hash listings

Bit Integrity Checker - Bulk

Description:

The Bulk Bit Integrity Checker provides a simple way to determine checksums (MD5s) for all content items in any particular space by leveraging an Amazon Hadoop cluster. This service is designed for large datasets (+10GB).

Configuration Options:

1. Space to verify: DuraCloud space where source files are stored
2. Service Mode
    a. Verify integrity of a Space: Retrieves all items in a space, computes the checksum of each, and compares that value with the MD5 value available from the storage provider
    b. Verify integrity from an item list: Retrieves all items listed in the item list, computes the checksum of each, and compares that value with the MD5 value provided in the item list
        i. Space with input listing: The DuraCloud space in which the input listing file resides
        ii. Input listing name: Name of the content item which contains the listing of items over which to run the service
3. Standard vs. Advanced configuration
    a. Standard mode automatically sets up the service to be run
    b. Advanced mode allows the user to configure the number and type of servers that will be used to run the job
        i. Number of Server Instances: The number of servers to use to perform the duplication task.
        ii. Type of Server: The type (size) of server used as perform the task. The larger the server, the faster the processing will occur. Larger servers also cost more than smaller servers to run. For more information, see the Amazon EC2 documentation.

Service Ouputs
All outputs of this service are placed in the system space, *x-service-out*.

1. bitIntegrity-bulk/bitIntegrity-report-<date>.csv
    - Final report with status of integrity check
2. bitIntegrity-bulk/bitIntegrity-results.csv
    - Interim listing with hashes recalculated from content streams

Image Transformer

Description:

The Image Transformer provides a simple way to transform relatively small numbers of image files from one format to another.

Note that the ImageMagick service must be deployed prior to using the Image Transformer

Configuration Options:

1. Source Space: DuraCloud space where source image files are stored
2. Destination Space: DuraCloud space where transformed image files will be placed, along with a file which details the results of the conversion process
3. Destination Format: The image format to which the source files will be transformed
4. Destination Color Space: The colorspace of the transformed files, either "Source Image Color Space", meaning that the colorspace of the original image will be used, or sRGB, meaning that the colorspace will be transformed to sRGB.
5. Source file name prefix: Only files beginning with the value provided here will be transformed. For example, if you enter ABC, only files whose names begin with the string ABC will be processed. This field is optional.
6. Source file name suffix: Only files ending with the value provided here will be transformed. For example, you enter .jpg, only files whose names ends with the string .jpg will be processed. This field is optional.

Service Output
All outputs of this service are placed in the system space, *x-service-out*.

1. image-transformer/image-transformer-results-<date>.csv
    - Final report indicating images converted and any errors encountered.

Image Transformer - Bulk

Description:

The Bulk Image Transformer provides a simple way to transform image files from one format to another in bulk. This service uses Amazon's Elastic Map Reduce capability to run the image transformation task within a Hadoop cluster.

Configuration Options:

1. Source Space: DuraCloud space where source image files are stored
2. Destination Space: DuraCloud space where transformed image files will be placed, along with a file which details the results of the transformation process
3. Destination Format: The image format to which the source files will be transformed
4. Destination Color Space: The colorspace of the transformed files, either "Source Image Color Space", meaning that the colorspace of the original image will be used, or sRGB, meaning that the colorspace will be transformed to sRGB.
5. Source file name prefix: Only files beginning with the value provided here will be transformed. For example, if you enter ABC, only files whose names begin with the string ABC will be processed. This field is optional.
6. Source file name suffix: Only files ending with the value provided here will be transformed. For example, you enter .jpg, only files whose names ends with the string .jpg will be processed. This field is optional.
7. Standard vs. Advanced configuration
    a. Standard mode automatically sets up the service to be run
    b. Advanced mode allows the user to configure the number and type of servers that will be used to run the job
        i. Number of Server Instances: The number of servers to use to perform the duplication task.
        ii. Type of Server: The type (size) of server used as perform the task. The larger the server, the faster the processing will occur. Larger servers also cost more than smaller servers to run. For more information, see the Amazon EC2 documentation.

Service Output
All outputs of this service are placed in the system space, *x-service-out*.

1. image-transformer-bulk/image-transformer-results-<date>.csv
    * Final report indicating images converted and any errors encountered.

Building DuraCloud Software from Source

Introduction

DuraCloud application software is composed of many parts. A breakdown of the primary pieces is as follows:

* DuraStore - this web application provides the access to and management of storage resources, which includes handling the storage portion of the DuraCloud REST API
* StorageProviders - this set is made up of the StorageProvider interfaces and the implementations which connect to distinct cloud stores (currently Amazon S3, Rackspace CloudFiles, and EMC Atmos)
* DuraService - this web application handles the deployment and management of services within DuraCloud, which includes handling the services portion of the DuraCloud REST API
* Services - the set of all deployable services, as well as the support projects that allow the DuraCloud services infrastructure to function
* ComputeProviders - this set is made up of the ComputeProvider interfaces and the implementation which connect to distinct cloud compute services (currently Amazon EC2, using the typica library)
* Security - handles security for the DuraCloud applications
* Common - a set of projects which provide utilities for other portions of the codebase to reuse

The DuraCloud software, by its very nature, is designed to be integrated with underlying cloud storage and compute providers.
As may be expected, these integrations are exercised through many "integration tests". In order for DuraCloud to connect to these underlying providers, appropriate credentials must first be established and available to the build framework.
However, in order to get up and running quickly with the software, the baseline can be initially built with just its unit tests.
Once the developer has independently acquired credentials for one or more of the supported underlying storage providers (below) builds may then be run with integration tests.

* Amazon S3
* Rackspace CloudFiles
* Windows Azure

This guide lays out the three tiers of building/testing the baseline:

1. build with unit tests
2. build with storage and services integration tests
3. build with OSGi services container integration tests

Although this document is written from a Linux environment perspective, analogous builds/installations have been tested in Windows (but may have limitations, as noted below). Any comments or feedback are welcomed.

Prerequisites

Software that must be installed on your system prior to building/using DuraCloud

1. Maven 2.2.1 or above
2. Tomcat 6.x or above
3. Java 6 (note: the djatoka service has compatibility issues with open-jdk)
4. Subversion

Building DuraCloud

> ℹ️ Any portions of the configuration below for which you need to include a replacement value will be written in all capital letters and included in brackets: [LIKE-THIS]

Build with unit tests

1. Check out latest stable release from Subversion repository

   ```
   svn co https://svn.duraspace.org/duracloud/tags/duracloud-1.0.0
   ```

2. Set environment variables

   ```
   export JAVA_OPTS="-XX:MaxPermSize=256m"
   export MAVEN_OPTS="-Xmx1024m -XX:MaxPermSize=1024m"
   ```

3. Configure Tomcat
   a. Add to $CATALINA_HOME/conf/tomcat-users.xml

   ```
   <tomcat-users>
     <role rolename="manager"/>
     <role rolename="admin"/>
     <user username="[ANY-USERNAME]" password="[ANY-PASSWORD]" roles="admin,manager"/>
   </tomcat-users>
   ```

4. Start tomcat

   ```
   $CATALINA_HOME/bin/startup.sh
   ```

5. Configure Maven2
   a. Add tomcat user to $M2_HOME/conf/settings.xml

   ```
   <servers>
     <server>
       <id>tomcat-server</id>
       <username>[ANY-USERNAME]</username>
       <password>[ANY-PASSWORD]</password>
     </server>
   </servers>
   ```

6. Build
   a. From top of source tree

```
mvn clean install
```

Note: Due to the length of time required to execute integration tests as well as the nature of "eventually consistent" cloud services to not cooperate with a synchronous test suite, these tests are not included in the build by default.

Build with OSGi services container integration tests

This step assumes the successful completion of the previous build instructions

<table>
<tr><th>Linux/Mac</th></tr>
<tr><td>

1. Start OSGi service container

```
cd //services/servicesadmin
mvn clean -f pom-run.xml pax:provision
cd runner
chmod +x run.sh
export BUNDLE_HOME=[DURACLOUD_HOME]/osgi-container
./run.sh
```

    a. Where [DURACLOUD_HOME] is a directory where the application has write access (can be same as <duracloud.home> set in Maven settings.xml above)
    b. The run.sh script will start an OSGi container and commandline interface to it
    c. The container starts with required bundles including the 'services-admin' installed
    d. See discussion below on OSGi container for more details
2. Once the 'services-admin' is running, tests that deploy services into the OSGi environment may be run
3. From inside the //integration-test module

```
mvn install -PrunServicesAdminTests
```

</td></tr>
</table>

Windows

1. Set up OSGi service container

```
cd services/servicesadmin
mvn clean -f pom-run.xml pax:provision
cd runner
```

2. (Optional) Set the OSGi bundle storage location

```
set BUNDLE_HOME=[BUNDLE_HOME]
```

   a. Where [BUNDLE_HOME] is the full path to an empty directory where the osgi container content will be stored
   b. Open the run.bat file in the runner directory in a text editor and replace all instances of "$BUNDLE_HOME" with "%BUNDLE_HOME%"
   c. Note: A directory called "$BUNDLE_HOME" under the runner directory will be used as the default bundle home if one is not specified.
3. (Optional) Set up logging
   a. Download the logback.xml file found here into your bundle home directory.
   b. Open the logback.xml file in a text editor and edit the LOG_FILENAME property to point to a full file path (including file name) for a log file.
   c. Note: One benefit to performing this step will be faster start time for your OSGi container.
4. Start OSGI service container

```
run.bat
```

   a. The run.bat script will start an OSGi container and commandline interface to it
   b. The container starts with required bundles including the 'services-admin' installed
   c. See discussion below on OSGi container for more details
5. Once the 'services-admin' is running, check to ensure that it was created properly
   a. In the console where run.bat was executed, an "osgi" prompt should be available. If it is not available, hitting enter should bring it up.
   b. Type "ss" and hit enter. This should list all of the available bundles. This list should include 50 items, all of which are either in the ACTIVE or RESOLVED state.

Optional items

Code coverage

1. If you plan on using Clover, the following element needs to be added to your maven 'settings.xml'

```
<profiles>
  <profile>
    <id>profile-clover</id>
    <activation>
      <property>
        <name>profile</name>
        <value>clover</value>
      </property>
    </activation>
    <properties>
      <cloverLicense>[LOCATION-OF-clover.license-FILE]</cloverLicense>
    </properties>
  </profile>
</profiles>
```

2. To run clover

```
mvn clover2:instrument clover2:aggregate clover2:clover -Pprofile-clover
```

3. A report will be generated in the following directory:
   //target/site/clover/

Logging

1. DuraCloud uses the SLF4j logging framework backed by the LogBack implementation
2. By adding either a logback.xml or logback-test.xml file on the classpath, logging configuration can be customized

DuraCloud internal tools

ServicesAdmin CLI

1. This tool provides a commandline interface for interacting with the 'services-admin' installed in a running OSGi container (see notes above for starting the container)
2. To build and run the CLI, from within the //servicesadminclient module

```
mvn assembly:assembly
java -cp target/servicesadminclient-[VERSION]-cli.jar
```

Application initialization utility

1. This utility takes a config file (example at //app-config/src/main/resources/init.props) and initializes an instance of duracloud
2. Until the applications durastore and duraservice are initialized, they are non-functional
3. To build and run the app-config utility, from within the //app-config module

```
mvn assembly:assembly
java -jar target/app-config-1.0.0-driver.jar
```

StoreClient package

1. To create a distributable zip of the storeclient and its dependencies, from within //storeclient run

```
mvn install -Ppackage-client
```

2. The zip will be found at /storeclient/target/store-client.zip

Misc configuration/discussion

Services on Windows

The following services do not function in a Windows deployment environment

- WebAppUtilService
- HelloWebappWrapper
- J2KService
- ImageMagickService

If you would like to run the ImageConversionService, you must install ImageMagick and have its /bin directory in your PATH, which is essentially what the ImageMagickService does in a linux environment.

root user

application config

OSGi container

DuraCloud Features

- **Application Features**
  - Storage system REST API
    - Space (top level folder) listing
    - Space creation

- Space deletion
- Space metadata and tagging
- Content (file) listing
- Content storage
- Content deletion
- Content metadata and tagging
- Service administration REST API
  - Available service listing
  - Deployed service listing
  - Service deployment
  - Service configuration
  - Service undeployment
- Reporting system REST API
  - Storage report listing
  - Storage report retrieval
  - Storage report starting and stopping
  - Storage report scheduling
  - Service report listing
  - Service report retrieval
- Security requiring authentication on all DuraCloud applications
- Web user interface (DurAdmin)
  - Access to all storage system capabilities, including space and content creation, updates, and deletes
  - Access to all service administration capabilities, including service deployment, configuration, and undeployment
  - Bulk deletion of spaces and content items
  - User account administration

- **Core Services**
  - **Media Streamer:**
    - Provides streaming capabilities for video and audio files.
  - **Duplicate on Change:**
    - Provides a simple mechanism for synchronizing your content between two storage providers. A running Duplicate on Change service will notice when a content item or space is added, updated, or deleted in one store and duplicate those changes in another store.
  - **Duplicate on Demand:**
    - Provides a simple way to duplicate content (and its associated metadata) from a space in the primary storage provider to a space in another provider.
  - **Image Server:**
    - Deploys an instance of the Adore Djatoka web application which provides serving and viewing of JPEG2000 images. Note that in order to view images using the Image Server, the images must be in an OPEN space.
  - **Image Transformer:**
    - Provides a simple way to convert image files from one format to another.
  - **Image Transformer - Bulk:**
    - Handles the conversion of large numbers of images files from one format into other.
  - **Bit Integrity Checker:**
    - Provides the ability to verify that the content held within DuraCloud has maintained its bit integrity.
  - **Bit Integrity Checker - Bulk:**
    - Provides a simple way to determine checksums (MD5s) for all content items in any particular space by leveraging an Amazon Hadoop cluster. This service is designed for large datasets (+10GB).
  - **Bit Integrity Checker - Tools:**
    - Provides additional bit integrity checking utilities which can be used to perform specific integrity checking tasks.

- **Tools**
  - **Sync Tool:**
    - Provides a command line utility for keeping DuraCloud content synchronized with the local file system.
  - **Retrieval Tool:**
    - Provides a command line utility for transferring content stored in DuraCloud to the local file system.
  - **Chunker Tool:**
    - Provides a command line utility for transferring single files to DuraCloud.

DuraCloud Security

Overview

The security approach is divided into two distinct spheres of responsibility

1. Channel security (encryption)
2. Application security (AuthN / AuthZ)

The configuration of any given user compute instance will consist of an Apache HttpServer layered on top of Tomcat.

1. Apache HttpServer
   - All requests will come through Apache on port 443 (https) of the instance
   - The requests will internally be unencrypted, where encryption exists, and redirected to tomcat as open text
2. Tomcat
   - A defined set of resource endpoints will require AuthN and AuthZ
   - Spring-security is being leveraged to wire AuthN and AuthZ across relevant resources

Channel Security Implementation

1. Apache HttpServer is configured to require all requests to the three DuraCloud web applications (/duradmin, /durastore, and /duraservice) go over https.
2. Below are the https enforcement rules configured in Apache

```
###
  # ensure 'duradmin' uses https
  ###

  RewriteCond %{REQUEST_URI} /duradmin
  RewriteCond %{SERVER_PORT} !^443$
  RewriteRule ^(.*)$ https://%{SERVER_NAME}$1 [R=301,L]

  ###
  # require https for 'durastore' & 'duraservice' for external requests
  ###

  RewriteCond %{REQUEST_URI} ^(/durastore|/duraservice)
  RewriteCond %{SERVER_PORT} !^443$
  RewriteCond %{SERVER_NAME} !^localhost$
  RewriteCond %{SERVER_NAME} !^127.0.0.1$
  RewriteCond %{REMOTE_HOST} !^127.0.0.1$
  RewriteCond ${local-ip-map:%{REMOTE_HOST}} !^localhost$
  RewriteRule ^(.*)$ https://%{SERVER_NAME}$1 [R=301,L]
```

Application Security Implementation

The basic AuthN flow is as follows

1. User requests secured resource
2. If credentials not in request
   - response 401
3. Spring AuthenticationProvider performs AuthN
   a. AuthProvider asks UserDetailsService for GrantedAuthorities for given Principal
   b. notes
      i. DuraCloud provides custom UserDetailsService implementation to return UserDetails of requesting Principal
      ii. AbstractSecurityInterceptor permanently caches user AuthN decisions by default
4. Authentication object and "configuration attributes" are passed to AccessDecisionManager for AuthZ

Security Servlet Filters

DuraCloud leverages Spring's mechanism for wiring AuthN/Z into an application across servlet url patterns.
The following access rules are placed across the durastore and duraservice REST-APIs:

Store REST Methods

Action	Role
Initialize Stores	ROLE_ROOT
Initialize Security Users	ROLE_ROOT
Get Stores	ROLE_USER
Get Spaces	ROLE_ANONYMOUS if space 'open', else ROLE_USER
Get Space	ROLE_ANONYMOUS if space 'open', else ROLE_USER
Get Space Metadata	ROLE_ANONYMOUS if space 'open', else ROLE_USER
Create Space	ROLE_USER
Set Space Metadata	ROLE_USER
Delete Space	ROLE_USER
Get Content	ROLE_ANONYMOUS if space 'open', else ROLE_USER
Get Content Metadata	ROLE_ANONYMOUS if space 'open', else ROLE_USER
Store Content	ROLE_USER
Set Content Metadata	ROLE_USER
Delete Content	ROLE_USER

Service REST Methods

Action	Role
Initialize Services	ROLE_ROOT
Initialize Security Users	ROLE_ROOT
Get Services	ROLE_USER
Get Service	ROLE_USER
Get Deployed Service	ROLE_USER
Get Deployed Service Properties	ROLE_USER
Deploy Service	ROLE_USER
Update Service Configuration	ROLE_USER
UnDeploy Service	ROLE_USER

Report UI REST Methods

Action	Role
Initialize DuraReport Application	ROLE_ROOT
Initialize Security Users	ROLE_ROOT
Get Latest Storage Report	ROLE_USER
Get Storage Report List	ROLE_USER
Get Storage Report	ROLE_USER
Get Storage Report Info	ROLE_USER
Start Storage Report	ROLE_USER
Cancel Storage Report	ROLE_USER
Schedule Storage Report	ROLE_USER
Cancel Storage Report Schedule	ROLE_USER
Get Deployed Services Report	ROLE_USER
Get Completed Services Report	ROLE_USER
Get Completed Services Report List	ROLE_USER
Get Services Report	ROLE_USER

Administrative UI REST Methods

Action	Role
Initialize DurAdmin Application	ROLE_ROOT
Initialize Security Users	ROLE_ROOT

Roles

The fixed set of users/roles listed below are provided in DuraCloud. Each role in the list below represents a super set of the privileges of those above it.

1. ROLE_ANONYMOUS
   - no username/password
2. ROLE_USER
   - user created by DuraCloud-account admin
3. ROLE_ADMIN
   - owner of DuraCloud-account
4. ROLE_SYSTEM
   - internal user for delegation requests
5. ROLE_ROOT
   - DuraSpace personnel

User Management

1. In order for the administrator of a DuraCloud account to manage new users, an initial user with ROLE_ADMIN privileges is provided at start-up:
   - username: admin
   - password: changeme

Note: *Ultimately, the management of users will take place through the DuraCloud.org website (where users initially create accounts).*
User management is exposed in Duradmin in the upper-righthand corner of the console, but the usernames/passwords edited here are not persisted.
This means that when the application is migrated to the next release, those details will need to be re-entered/re-created.

Service Development Guide

Introduction

This document is intended to provide guidelines for developing your own custom services that can be deployed into the DuraCloud services framework.
The framework has been designed to allow any

- Java code that can be
- deployed as an OSGi bundle and
- implements the ComputeService interface
  to be dynamically installable into a running DuraCloud application instance.

Overview

In order to understand the flow of services within the services framework, there are several entities with which to become familiar:

1. ***DuraService***

   - web application that exposes a REST-API for end-user management of services
     a. deploy service
     b. undeploy service
     c. configure service
     d. get service listing
     e. get service properties
   - responsible for retrieving a service to be deployed from the **Service Registry** and streaming it to the **ServicesAdmin**
   - responsible for mediating subsequent end-user management of deployed services to **ServicesAdmin**
   - responsible for passing metadata to the OSGi container when a service is being deployed.
     - metadata is defined in the service-repo.xml which is built based on the ServiceInfo class for each service

2. ***Service Registry***

   - persistent holding area of deployable services
   - contains **Service Packages** and a single services-configuration descriptor XML object
   - implemented as one or more DuraCloud "space(s)"
     - the properties file to initialize DuraCloud contains duraservice.service-storage properties which allow configuration of the host, port, context and space

3. ***Service Package***

   - for simple services, this can be a single OSGi bundle, jar file
   - for more involved services, this would be a zip file containing
     a. OSGi bundle jar that implements the ComputeService interface
     b. OSGi bundle jars that are both dependencies of the ComputeService bundle (above) and not among the base bundles installed in the **OSGi Container** by default
     c. any support files (see 'Example Services' below)

4. ***ServicesAdmin***

   - management component resident in the OSGi container
   - exposes an internal REST-API through which communication from **DuraService** takes place
   - responsible for actually communicating with services via their common ComputeService interfaces
   - responsible for keeping track of currently deployed services and dependency-link counters

5. ***OSGi Container***

   - production implementation currently in use is Equinox
   - bundles installed by default are listed as dependencies in the pom-run.xml file in the ServicesAdmin project

Example Services

All of the below services are currently functional and available in the DuraCloud source baseline.
They can all be found within the services directory

Pure Java Services

1. Hello Service
2. Replication Service
3. WebAppUtil Service
4. ImageConversion Service
5. MediaStreaming Service

Web Application Services

1. HelloWebAppWrapper Service
2. J2K Service

System Services

1. Script Service
2. ImageMagick Service

General Compute Services

Implementation Guidelines

OSGi Compatibility

- Although there are several available OSGi containers, and the DuraCloud test harnesses employ Equinox, Felix, and Knoplerfish, the production container into which services need to be able to deploy is **Equinox**.

ComputeService Interface

Dependency Packaging

Service Descriptor

DuraCloud Retrieval Tool

Introduction

The Retrieval Tool is a utility which is used to transfer (or "retrieve") digital content from DuraCloud to your local file system.

Download

Download the retrieval tool from the Downloads page.

How the Retrieval Tool Works

- When the Retrieval Tool starts up, it connects to DuraCloud using the connection parameters you provide and gets a list of content items in the spaces you indicate. It will then proceed to download the files from those spaces, each into a local directory named for the space, which is placed within the content directory.
- For each content item, the Retrieval Tool checks to see if there is already a local file with the same name. If so, the checksums of the two files are compared to determine if the local file is the same as the file in DuraCloud. If they match, nothing is done, and the Retrieval Tool moves on to the next file. If they do not match, the file from DuraCloud is retrieved.
- By default, when a local file exists and differs from the DuraCloud copy, the local file is renamed prior to the DuraCloud file being retrieved. If you would prefer that the local file simply be overwritten, you will need to include the overwrite command-line flag when starting the Retrieval Tool.
- As each content file is downloaded, a checksum comparison is made to ensure that the downloaded file matches the file in DuraCloud. If the checksums do not match, the file is downloaded again. This re-download will occur up to 5 times. If the checksums still do not match after the fifth attempt, a failure is indicated in the output file.
- As each file download completes, a new line is added to the retrieval tool output file in the work directory, indicating whether the download was successful or not. Files which did not change are not included in the output file.
- As the Retrieval Tool runs, it will print its status approximately every 10 minutes to indicate how many files have been checked and downloaded.
- Once all files are retrieved, the Retrieval Tool will print its final status to the command line and exit.
- As files are updated in DuraCloud, you can re-run the Retrieval Tool using the same content directory, and only the files which have been

added or updated since the last run of the tool will be downloaded.

Operational notes

- Content Directory - the directory to which files will be downloaded. A new directory within the content directory will be created for each space.
- Work Directory - the work directory contains both logs, which give granular information about the process, and output files. A new output file is createdc for each run of the Retrieval Tool, and it stores a listing of the files which were downloaded.

Prerequisites

- You must have Java version 6 or above installed on your local system. If Java is not installed, you will need to download and install it. To determine if the correct version of Java is installed, open a terminal or command prompt and enter

```
java -version
```

The version displayed should be 1.6.0 or above. If running this command generates an error, Java is likely not installed.
- You must have downloaded the Retrieval Tool. It is available as a link near the top of this page.

Starting the Retrieval Tool

- To run the Retrieval Tool, open a terminal or command prompt and navigate to the directory where the Retrieval Tool jar file is located
- To display the help for the Retrieval Tool, run

```
java -jar retrievaltool-[version]-driver.jar
```

- When running the Retrieval Tool, you will need to use these options:

Short Option	Long Option	Argument Expected	Required	Description	Default Value (if optional)
-h	--host	Yes	Yes	The host address of the DuraCloud DuraStore application	
-r	--port	Yes	No	The port of the DuraCloud DuraStore application	443
-u	--username	Yes	Yes	The username necessary to perform writes to DuraStore	
-p	--password	Yes	Yes	The password necessary to perform writes to DuraStore	
-i	--store-id	Yes	No	The Store ID for the DuraCloud storage provider	The default store is used
-s	--spaces	Yes	No	The space or spaces from which content will be retrieved. Either this option or -a must be included	
-a	--all-spaces	No	No	Indicates that all spaces should be retrieved; if this option is included the -s option is ignored	Not set
-c	--content-dir	Yes	Yes	Retrieved content is stored in this local directory	
-w	--work-dir	Yes	Yes	Logs and output files will be stored in the work directory	
-o	--overwrite	No	No	Indicates that existing local files which differ from files in DuraCloud under the same path and name sould be overwritten rather than copied	Not set
-t	--threads	Yes	No	The number of threads in the pool used to manage file transfers	3

- An example for running the Retrieval Tool

```
java -jar retrievaltool-1.0.0-driver.jar -c content -h test.duracloud.org -u myname -p
mypassword -w work -s space1 space1 -o
```

Known Issues

The following issues are known to exist in release 1.0.0 of DuraCloud:

Title	Issue	Tracker Item	Work Around
Incorrect redirects from HTTP to HTTPS	When performing PUT/POST/DELETE requests via the REST API, if the URL uses http:// rather than https:// the response is a 404.	DURACLOUD-255	Use https:// when performing requests via the REST API of hosted DuraCloud instances
Login requires multiple attempts	When logging into DurAdmin, the first time a username/password is entered and submitted, the fields simply clear. A second attempt to login with the same credentials succeeds.	DURACLOUD-260	If the initial login attempt fails, attempt to log in again.
Incorrect MIME types set on files	When uploading files via the sync tool which do not have an extension, the MIME type is always set to application/octet-stream	DURACLOUD-227	Set the MIME type of these files either through DurAdmin or via the REST API
File not streaming with Media Streamer deployed	A file (or set of files) do not get permissions set properly to allow streaming	DURACLOUD-335	Redeploy the Media Streamer with the same settings.
Bulk Image Transformer converting from JP2	The Bulk Image Transformer has seen failures when converting JP2 files into other formats	DURACLOUD-345	Use a larger type and/or number of servers for this type of job

More issues and planned improvements can be found on the DuraCloud issue tracker.