DuraCloud Release 3.6

Getting Started

- Getting Started with DuraCloud
- How to Automatically Upload Files and Directories to DuraCloud

User Documentation

- Release Notes
- Known Issues
- DuraCloud Features
- DuraCloud Storage
- DuraCloud Media Streaming
- DuraCloud REST API
- DuraCloud Security
- DuraCloud Java Clients
- DuraCloud Sync Tool
- DuraCloud SyncOptimize Tool
- DuraCloud Retrieval Tool
- DuraCloud Chunker Tool
- DuraCloud Stitcher Tool
- DuraCloud Administration

Developer Documentation

- Deploying DuraCloud from Binaries
- Building DuraCloud from Source
- Logging Configuration

Release Notes

Release 3.6.0

Released: November 23, 2015

The DuraCloud 3.6.0 release added features and updates to

- Administrators now see a "Request Restore" button on completed snapshots. Clicking this button sends DuraCloud staff an email and logs the event in the snapshot history.
- All snapshots now have a "Preservation Network Member ID" field in the snapshot details, which captures the DPN Member ID.
- When listing snapshots through the bridge REST API there are now several optional parameters to limit results.
- Restore actions (requesting, initiating, completing, expiring) are automatically included in the snapshot history.
- Non-administrative users can now view snapshots based on the permissions assigned to the associated space. So if a user has access to a space, they also have access to see the snapshots taken of that space.
- Performing multiple snapshots at once is now supported.

To see the full list of changes, or for more details on specific changes in release 3.6.0, see the JIRA issue tracker.

Release 3.5.0

Released: October 19, 2015

The DuraCloud 3.5.0 release focuses on smoothing the integration with DPN. The key features that were added that are visible in DuraCloud:

- Snapshot details, history, and artifacts are downloadable.
- Manifest and audit logs are now visible to anyone with access to a space.
- "Single Pass" mode for users running the DuraCloudSync tool in the GUI mode.
- A snapshot action can now be restarted or cancelled, and any errors that occur while performing a snapshot can be reported properly.
- Improved handling of chunked large files in snapshots.

To see the full list of changes and bug fixes, or for more details about specific changes in release 3.5.0, see the DuraCloud Sprint and DuraCloud Vault Sprint

Release 3.3.0

Released: June 15, 2015

The DuraCloud 3.3.0 release adds two primary features:

- Secure media streaming: There is now the option to stream audio and video files from a DuraCloud space using either open or secure streaming. Open streaming provides consistent URLs for streamed content items, which is ideal for open access data, as anyone with the URL can stream the content. Secure streaming, in contrast, allows a signed URL to be generated each time a file is to be streamed. These signed URLs can be used for only a certain amount of time, and can also be limited to a particular IP range. Secure streaming allows for streaming of media content to a limited audience.
- Two factor authentication: DuraCloud will now accept an IP address, IP range, or list of IP ranges from which requests must originate for a given user to successfully log in to their DuraCloud account. This adds a second layer of security by disallowing any attempts to log in to DuraCloud which do not come from the expected internet address. This can be used, for example, to ensure that attempts to log in to a DuraCloud account always occur from an on-campus location.

Beyond the two primary security-focused features, many updates were made to further support the DuraCloud Mill and to resolve a variety of bugs.

To see the full list of changes, or for more details about specific changes in release 3.3.0, see the JIRA issue tracker.

Release 3.2.0

Released: December 17, 2014

The DuraCloud 3.2.0 release is focused on two main themes: Integration with DPN, and integration with the DuraCloud Mill.

- DPN Integration: A new Snapshot Storage Provider was added to DuraCloud, which is used to integrate with Chronopolis, allowing DuraCloud and Chronopolis to function together as a single DPN node.
- DuraCloud Mill Integration: The back end of DuraCloud, where all of the bits are processed, has moved into system called the DuraCloud Mill. This system manages the audit processing, manifest generation, duplication, and bit integrity functions of the DuraCloud hosted service. This version of DuraCloud transitioned many of those functions to the Mill, and added REST API and UI components which allow users to interact with the Mill to retrieve things like space audit logs and manifest files.

Beyond the two major themes, this release includes significant housekeeping work as well. The full set of project dependencies were reviewed, updated, and significantly pruned. Each piece of the DuraCloud software was reviewed to ensure its configuration and dependency set were up-to-date and correct. Much of this was in preparation for the DuraCloud software being hosted by Sonatype, which means that the 3.2.0 release is the first DuraCloud release to be available in Maven Central. This makes the DuraCloud software much easier to reuse in other Java projects.

Also notable is a new "jump start" feature in the DuraCloud SyncTool which enables users to dramatically increase their throughput for first time uploads of large numbers of small files. See DuraCloud Sync Tool or DuraCloud Sync Tool - Command Line for details.

To see the list of housekeeping tasks and bug fixes in release 3.2.0, see the JIRA issue tracker. The work completed on the DuraCloud Mill is maintained in a stand-alone JIRA project. The work on the Chronopolis integration is also in a stand-alone JIRA project.

Release 3.1.0

Released: June 6, 2014

The primary features of release 3.1.0 are:

- The DuraCloud SyncOptimize Tool, a new tool that can be accessed both as a command-line tool as well as through the Sync Tool User interface, helps you optimize throughput from your local machine when uploading content to DuraCloud.
- The Sync Tool UI now enables users to manually set the thread count to improve throughput performance.
- The Sync Tool now supports the use of a prefix. If a prefix is used, that value is added to the beginning of all content IDs created in DuraCloud by the Sync Tool. For details on how it works, see DuraCloud Sync Tool orDuraCloud Sync Tool - Command Line.

To see the full list of changes, or for more details about specific changes in release 3.1.0, see the JIRA issue tracker.

Release 3.0.0

Released: April 22, 2014

The DuraCloud 3.0.0 release is the first step towards allowing DuraCloud to be run in a more distributed and efficient manner. Towards that end, the services framework has been removed in favor of handling the same tasks (primarily duplication between providers and bit integrity checking) through a scalable processing engine which is run outside of the DuraCloud instances themselves. This provides for better performance on the instances and better performance for the services, as well as less effort for DuraCloud users.

The primary features of release 3.0.0 are:

- Users of DuraCloud no longer need to manage services. The service framework has been replaced by a back-end task processing system that leverages the massively parallel processing capabilities of the cloud.
- Library updates, to keep the primary components of DuraCloud software up-to-date. Examples: Upgraded from Maven 2 build system to Maven 3. Upgraded the Spring Framework version from 2.5 to 4.0.2.
- The auditing framework's use of ActiveMQ have been removed in favor of a more pluggable architecture using an external queuing system. Audit event logging is now being handled by the same task processing framework that is managing duplication and bit integrity checks.
- Media streaming capabilities updated to continue to work just as before, but without the need for the Media Streamer service.

To see the full list of changes, or for more details about specific changes in release 3.0.0, see the JIRA issue tracker.

Release 2.4.0

Released: September 20, 2013

The primary features of release 2.4.0 are:

- New drag and drop capability for file upload
    - DuraCloud now provides the option to upload files using the familiar file selector or with simple drag and drop.
        - This new functionality is enabled using HTML5, which removes the need for a java browser plugin.
- New installers for the Sync Tool
    - The Sync Tool can now be installed using a platform-specific installer on Windows, Mac, and Linux.
    - The installers provide a simple graphical installation flow and are consistent with the usual installation process on each operating system.
    - The installers provide the option to start up the Sync Tool on system restart, to ensure that the Sync Tool will continue to run even if you need to reboot your computer.
    - The installers add links and shortcuts which make it easy to find the Sync Tool when you need to see progress or change settings.
- New features for the Sync Tool
    - The Sync Tool now provides 3 options for handling files which have changed locally:
        1. Overwrite the file (the default, and the only option available in previous versions)
        2. Skip the file (ensures that there are no changes made to any existing files in DuraCloud)
        3. Rename original (rename the old copy of this file first, then add the new file to DuraCloud)
    - The Sync Tool now allows the DuraCloud password to be provided via a system property or via a prompt, both of which remove the need to include a password as part of the command line call
- New features for the Retrieval Tool
    - The Retrieval Tool can now retrieve a list of the files in a space
    - The Retrieval Tool can now retrieve a specific set of files from a space, as well as being able to retrieval all files in a space
    - The Retrieval Tool now allows the DuraCloud password to be provided via a system property or via a prompt, both of which remove the need to include a password as part of the command line call

To see the full list of changes, or for more details about specific changes in release 2.4.0, see the JIRA issue tracker.

Release 2.3.1

Released: March 28, 2013

Release 2.3.1 is a bug fix release which focused on resolving bugs which resulted in errors when uploading files via the Sync Tool.

For more details about specific changes in release 2.3.1, see the JIRA issue tracker.

Release 2.3.0

Released: March 1, 2013

The primary features of release 2.3.0 are:

- A new graphical Sync Tool
    - The Sync Tool can now be run using either a command line or a graphical user interface
        - The new graphical interface provides a setup wizard and an administration console with a monitoring display.
        - The familiar command line interface continues to provide access to all sync features, some of which are not yet available via the graphical interface.
        - Regardless of which interface is used, the underlying sync functionality remains the same.
    - The Sync Tool now captures and preserves the time stamps of files being transferred to DuraCloud, and the Retrieval Tool now re-instates those time stamps when files are retrieved.
    - The Sync Tool's command line interface now supports an exclude list, which can be used to specify files and directories that

should not be pushed to DuraCloud.
- Amazon Glacier integration
    - Amazon Glacier is now available as a secondary storage provider in DuraCloud

For more details about specific changes in release 2.3.0, see the JIRA issue tracker.

Important Known Issues:

- Java Browser Security Vulnerabilities
    - The DuraCloud Upload Tool, which is deployed as a Java browser applet to assist with file uploads through the DuraCloud web UI, requires the Java browser plugin be installed.
    - Due to known security vulnerabilities in Java browser plugins, it is recommended that users upgrade their local version of Java to the latest available.
    - If you prefer to not use a Java browser plugin, or if the plugin is not functioning properly, we recommend using the newly updated Sync Tool, which is available from the "Get Sync Tool" button in the DuraCloud web UI, or from the DuraCloud downloads page.

A detailed list of known issues in release 2.3.0 may be found found here.

Release 2.2.0

Released: November 14, 2012

The primary features of release 2.2.0 are:

- Java 7 support
    - The DuraCloud code base now requires Java 7 to build and run. This update is necessary both because Java 6 is reaching end of life, and to provide access to new features in Java 7.
    - All client-side Java tools now require Java 7 to run, this includes the Sync Tool, the Retrieval Tool, and the Upload Tool. Previous versions of these tools will continue to work with Java 6.
    - The file upload capability built into the DuraCloud UI (the Upload Tool in applet form) will now require Java browser plugins be updated to Java 7. Simply installing Java 7 for your platform should install the appropriate browser plugins.
- A long list of UI tweaks and improvements including better graph and report display and labeling, improved multi-select support, and a variety of other small updates.

For more details about specific changes in release 2.2.0, see the JIRA issue tracker

Release 2.1.1

Released: September 28, 2012

The primary features of release 2.1.1 are:

- Duplicate on Change service updates
    - Duplication settings for newly created spaces can now be specified using a default setting. Any new spaces created will be configured using the default duplication settings.
- Space counting
    - The space counting feature in the UI now captures the final count on a space, and will only re-run the count when asked.

For more details about specific changes in release 2.1.1, see the JIRA issue tracker.

Release 2.1.0

Released: August 9, 2012

The primary features of release 2.1.0 are:

- Duplicate on Change service updates
    - Duplication is now configured at the space level, allowing for much finer grained selection of which content will be automatically copied to a secondary provider, and which provider that will be.
- SDSC Storage Provider
    - The SDSC storage provider connection was tested and improved, in close cooperation with SDSC personnel, to ensure its readiness for production status.
- Security
    - The DuraCloud security configuration has been extended to only allow Administrative users to perform service execution.

For more details about specific changes in release 2.1.0, see the JIRA issue tracker.

Release 2.0.0

Released: April 17, 2012

The primary features of release 2.0.0 are:

- Integrated History Reports
    - Reports illustrating both current and historical views of the content stored in DuraCloud have now been integrated directly into the display for each individual space.
- Automated Service Execution
    - Bit Integrity Checks
        - Bit Integrity checking is now an automated function. Each content item in each space in each storage provider will be checked regularly to verify that it has not changed since it arrived in DuraCloud. No user intervention is required for this to occur. As each space is checked, the display will update to indicate that the bit integrity of the contained content has been verified.
    - Media Serving
        - Streaming content from DuraCloud now requires only a single button click. Each space now provides the option to turn on or off streaming, with no need to configure, deploy, or re-deploy a service.
- Audit Logging
    - Logging is now being generated and captured which tracks the events occurring within DuraCloud. This provides a permanent record of when content is added, updated, or removed.
- Manifest Generation
    - Building on the work of the audit log, a content manifest can now be requested for any space within DuraCloud. This manifest, which can be in multiple formats, describes the content that resides in the space the moment that the manifest is requested.

For more details about specific changes in release 2.0.0, see the JIRA issue tracker.

Release 1.3.1

Released: Jan 20, 2012

The primary features of release 1.3.1 are:

- Improved service reporting
    - The information within service reports can now be viewed in tabular form directly from within the service details area.
    - Items which are considered error cases that occur as part of service execution are now included in an independent error report that is available for viewing upon service completion.
- Simplified public access
    - Setting the contents of a space to be publicly viewable is now accomplished by simply granting read access to the "public" group.
- CloudSync service
    - The latest version of CloudSync, a utility for managing the movement of content between DuraCloud and a Fedora repository, can now be run as a service within DuraCloud

For more details about specific changes in release 1.3.1, see the JIRA issue tracker.

Release 1.3

Released: Dec 14, 2011

The primary features of release 1.3 are:

- Improved access control
    - Administrators can now define access control lists for each space, indicating which users and groups have read or write access to the content within that space.
    - Users now see only the spaces in their spaces listing which they have access to view. This includes all Open spaces, which remain available for public read access.
    - Users now only see options to perform add, edit, or delete actions in spaces where they have write permissions.
- Content copy across providers
    - Files stored in DuraCloud can now be easily copied individually to another storage provider via both the REST API and the web interface.
- Email notification on service completion
    - After a service in DuraCloud completes, an email is now sent to the user who launched the service, notifying them that the service has completed, and providing details about the results of the service.

For more details about specific changes in release 1.3, see the JIRA issue tracker.

Release 1.2

Released: Oct 30, 2011

The primary features of release 1.2 are:

- Upload Tool
  - Provides a graphical method for transferring large numbers of files to DuraCloud.
  - Allows for the selection of both files and folders for transfer, and presents a visual indication of the transfer progress.
  - Can be run from the DuraCloud UI by choosing the "Add Many Items" button when a space is selected.
  - Can also be download and run locally.
- SDSC Storage Provider
  - An initial beta release of the new storage provider integration for connecting to the SDSC Cloud storage system.

For more details about specific changes in release 1.2, see the JIRA issue tracker.

Release 1.1

Released: Sept 30, 2011

The primary features of release 1.1 are:

- Media Streamer service
  - Service now allows content from multiple spaces to be streamed.
  - Service now recognizes when new content is added to spaces which are being streamed and starts streaming on those files.
- Duplicate on Change service
  - Service updated to provide greater assurance of file transfer and produce an output report which details the actions of the service and the results of those actions.
- Content item copy and rename capability
  - New copy operation is able to copy, move, and rename content items within a space and between spaces.
- Stitch Tool
  - New utility for large files which were "chunked" (split into multiple small files) when placed into storage. This new utility will recombine all of the pieces of a file to re-produce the original file in local storage.
  - Stitch capabilities incorporated into the Retrieval Tool, allowing any content which has been "chunked" to be reconstituted on retrieval.
- UI updates
  - Improvements to the user interface in order to provide simpler and more useful feedback for actions being performed throughout the application.
- Properties
  - Use of the term 'properties' has replaced 'metadata' to describe the name/value pairs which can be associated with spaces and content. This update helps to clarify the purpose and capability of this attached information.
- Local service repositories
  - Service repositories can now reside within the same storage container as is used by DuraStore, allowing for simpler configuration of stand-alone DuraCloud instances.
- Initialization endpoint
  - New REST API endpoint (/init) for application initialization.
- Tools tab
  - New tools tab on the dashboard provides convenient links for downloading DuraCloud client tools.

For more details about specific changes in release 1.1, see the JIRA issue tracker.

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* configu ration 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*

Getting Started with DuraCloud

This four minute video will show you how to get up and running with DuraCloud quickly. You will learn how to upload, annotate, view, and delete content within a space.

Once you've mastered the lessons offered in this video, you may find it helpful to learn How to Automatically Upload Files and Directories to DuraCloud.

How to Automatically Upload Files and Directories to DuraCloud

Introduction

DuraCloud supports uploading files through file selection or drag-and-drop via the web-based DuraCloud administrative interface. However, this method requires you to initiate the upload for each file or set of files every time you would like to update them. Also, this web-based administrator does not allow you to upload whole directories at a time. Enter the DuraCloud Sync Tool. This application allows you to continuously copy files from any number of local folders to a DuraCloud space. As you add, update, and delete files locally these changes will be automatically propagated to the cloud. You can use the tool in two different modes: GUI mode or via a command-line interface.

Running in GUI Mode

To run in GUI Mode, you can download and install one of our conveniently-packaged, platform-specific installers. Once the installation is complete, the application will guide you through the setup. You will be up and running in five minutes. For more guidance, watch this three minute video that demonstrates how to install, configure and run the DuraCloud Sync Tool in GUI mode.

Command-Line Mode

Command-line mode is useful for those users running in a server environment, for those who want to run the Sync Tool in scripts, or for those who simply prefer a command line interface.

DuraCloud REST API

DuraCloud REST API methods:

Notes

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

Security Initialization REST Methods

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`
- Request Body: XML similar to:

```
<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.

Initialization REST Methods

Initialize Stores

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

```
<durastoreConfig>
  <storageAudit>
    <auditUsername>audit-username</auditUsername>
    <auditPassword>audit-password</auditPassword>
    <auditQueue>audit-queue-name</auditQueue>
    <auditLogSpaceId>audit-log-space-id</auditLogSpaceId>
  </storageAudit>
  <millDb>
    <name>db-name</name>
    <host>db-host</name>
    <port>db-port</name>
    <username>db-username</username>
    <password>db-password</password>
  </millDb>
  <storageProviderAccounts>
    <storageAcct ownerId='0' isPrimary='true'>
      <id>1</id>
      <storageProviderType>AMAZON_S3</storageProviderType>
      <storageProviderCredential>
        <username>username</username>
        <password>password</password>
      </storageProviderCredential>
      <storageProviderOptions>
        <!-- Provider specific details -->
      </storageProviderOptions>
    </storageAcct>
  </storageProviderAccounts>
</durastoreConfig>
```

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

Is Initialized

- Purpose: Performs a check to determine if the DuraStore application has been initialized
- Request: `GET https://host:port/durastore/init`
- Response Code: 200 (if the application has been initialized), 503 (if the application has NOT been initialized)
- Response Body: Text indicating whether initialization has occurred.

Store REST Methods

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>
```

Space REST Methods

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 properties
- 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 properties, example:

```
x-dura-meta-space-count: 65
x-dura-meta-space-created: Mon, 01 Jan 2000 08:00:00 EST
```

Get Space Properties

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

Get Space ACLs

- Purpose: Provides all space ACLs, with values of 'r' (read) and 'w' (read/write)
- Request: `HEAD https://host:port/durastore/acl/spaceID ? (storeID)`
- Response Code: 200 (on success)
- Response Headers: All available space ACLs, example:

```
x-dura-meta-acl-user0: WRITE
x-dura-meta-acl-user1: WRITE
x-dura-meta-acl-group-curators: READ
```

Create Space

- Purpose: Creates a new space
- Request: `PUT https://host:port/durastore/spaceID ? (storeID)`
- 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 ACLs

- Purpose: Updates the ACLs associated with a space
- Request: `POST https://host:port/durastore/acl/spaceID ? (storeID)`
- Request Headers: For 'user' ACLs the header prefix must be 'x-dura-meta-acl-' and for 'groups' the header prefix must be 'x-dura-meta-acl-group-'. Allowable values for ACL headers are: 'READ' and 'WRITE'.
  Example:

```
x-dura-meta-acl-user0: WRITE
x-dura-meta-acl-user1: WRITE
x-dura-meta-acl-group-curators: READ
```

- Response Code: 200 (on success)
- Response Body: "Space $spaceID ACLs 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 properties
- 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 properties, example:

```
Content-Type: text/plain
Content-Length: 5732
Content-MD5: 345670923478509747383920 2
ETag: 34567092347850974738392 02
x-dura-meta-content-name: Testing Content
x-dura-meta-content-owner: JSmith
```

Get Content Properties

- Purpose: Retrieves the properties 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: Properties about the content, example:

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

- Response Code:
  - 201 (on success)
  - 400 (if the content ID is invalid)
  - 404 (if the the given space does not exist)
  - 409 (if the provided checksum did not match the stored content checksum)
  - 500 (on error)
- Response Headers:
  - MD5 checksum of stored content
  - ETag of stored content
  - Location of the new content (i.e. the URL used to create the content), example:

```
Content-MD5: 4cd56e137a93a1accb43c5d32f4afffb
ETag: 4cd56e137a93a1accb43c5d32f4afffb
Location: https://myhost:8080/durastore/space1/content1
```

- Usage Notes
  - 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.

Copy Content

- Purpose: Copies a piece of content from a source space to a destination space within a given store
- Request: `PUT https://host:port/durastore/spaceID/contentID ? (storeID)`
- Request Body: must not exist
- Request Headers: Copy source, example:

```
x-dura-meta-copy-source: space-id/content-id
```

- Optional Request Headers: Copy source store, example:

```
x-dura-meta-copy-source-store: storeId
```

- Response Code: 201 (on success)
- Response Headers:
  - MD5 checksum of stored content
  - ETag of stored content
  - Location of the new content (i.e. the URL used to create the content), example:

```
Content-MD5: 4cd56e137a93a1accb43c5d32f4afffb
ETag: 4cd56e137a93a1accb43c5d32f4afffb
Location: https://myhost:8080/durastore/space1/content1
```

- Usage Notes
  - The properties associated with the source content item are copied to the destination content item.
  - The source and destination spaces may be the same.
  - Including the optional header indicates that the copy action should retrieve the source file from a space in the specified storage provider. This allows for copying a file from one storage provider to another.

Set Content Properties

- Purpose: Updates the properties associated with a piece of content. **Note:** You must include ALL properties you would like associated with the given content item in this call. Any properties that exist before this call but are not included in the call itself will be removed. This is to allow for both adding and removing properties.
- 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"

Audit Log REST Methods

Get Audit Log

- Purpose: Returns the latest audit for a given store and space
- Request: `GET https://host:port/durastore/audit/{spaceId} ? (storeID)`
    - `Optional parameter 'storeID': if not set, primary store storage provider is used.`
- Response Code: 200 (on success), 404 if audit logs were not found.
- Response Body: TSV in chronological order with the following fields.

```
ACCOUNT STORE_ID SPACE_ID CONTENT_ID CONTENT_MD5 CONTENT_SIZE
CONTENT_MIMETYPE CONTENT_PROPERTIES SPACE_ACLS SOURCE_SPACE_ID
SOURCE_CONTENT_ID TIMESTAMP ACTION USERNAME
mysubdomain 51 myspace image-01.jpg b1978f9fc4fe9448e05b83bbe6b98109
81214 image/jpeg {"content-mimetype" : "image/jpeg"}  {}
2014-09-10T15:54:42.042 ADD_CONTENT root
```

Manifest REST Methods

Get Manifest

- Purpose: Returns the manifest for a given space and storeId
- Request: `GET https://host:port/durastore/manifest/{spaceId} ? (storeID) (format)`
    - `Optional parameter 'storeID': if not set, primary store storage provider is used.`
    - `Optional parameter 'format': TSV or BAGIT. TSV is default.`
- Response Code: 200 (on success), 404 if manifest was empty found.
- Response Body: TSV in chronological order with the following fields.

TSV results

```
space-id content-id MD5
auditlogs
localhost/51/auditlogs/localhost_51_auditlogs-2014-09-10-15-56-07.tsv
6992f8e57dafb17335f766aa2acf5942
auditlogs
localhost/51/photos/localhost_51_photos-2014-09-10-15-55-01.tsv
820e786633fb495db447dc5d5cf0b2bd
```

Task REST Methods

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>
```

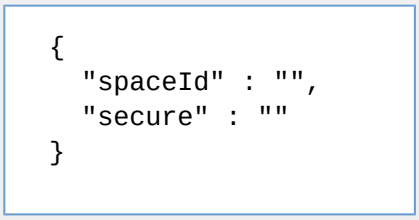
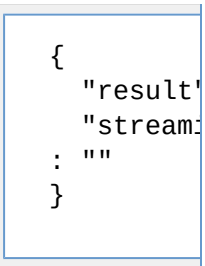
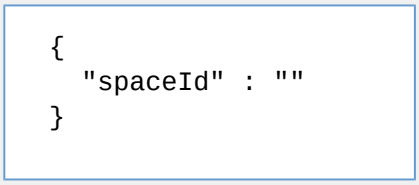
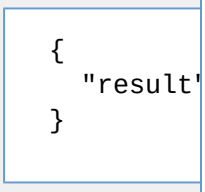
Perform Task

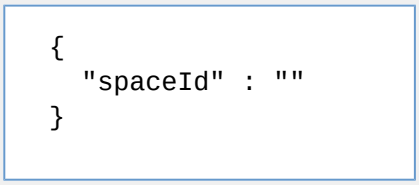
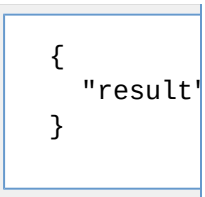
- Purpose: Performs a particular task. Note that most tasks can be performed by only one storage provider type.
- Request: `POST https://host:port/durastore/task/taskName ? (storeID)`
- Request Body: Parameters for task. Each task will expect parameters in a specific format, see task listing for more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

Tasks

Amazon S3 Storage Provider

taskName	Name	Description	Request Body	Response Body

enable-streaming	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. When this call completes, two new properties will have been added to the set of properties for the specified space: • streaming-host - this is the RTMP host value, which can be used to generate URLs for open streams • streaming-type - will either be OPEN or SECURE, depending on the value of the secure parameter provided when streaming was enabled	 spaceId - Name of the space for which streaming is to be enabled secure - true or false, should streaming be secured	 result - Text indicating the streamingHost - the host streaming endpoint
disable-streaming	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.	 spaceId - Name of the space for which streaming is to be disabled	 result - Text indicating the

delete-streaming	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.	 spaceId - Name of the space for which streaming is to be deleted	 result - Text indicating the
get-url	Get URL task	Retrieves a URL for a media file that is streamed through Cloudfront via an open distribution	 spaceId - Name of the space in which the streamed content is stored contentId - Name of the content item to be streamed resourcePrefix - A prefix on the content item which may be required by the streaming viewer. (e.g. an mp4 file may need a prefix of "mp4:") (optional)	 streamUrl - The URL to b the requested content

get-signed-url	Get Signed URL task	Retrieves a signed URL for a media file that is streamed through Cloudfront via a secure distribution	<pre>{
			"spaceId" : "",
			"contentId" : "",
			"minutesToExpire"
			: "",
			"ipAddress" : ""
			"resourcePrefix" :
			""
			}</pre>
			spaceId - Name of the space in which the streamed content is stored
			
			contentId - Name of the content item to be streamed
			
			minutesToExpire - Number of minutes until the generated URL expires and the stream can no longer be played (optional, default is 480)
			
			ipAddress - IP address range where requests to stream must originate, in CIDR notation (e.g. 1.2.3.4/32) (optional)
			
			resourcePrefix - A prefix on the content item which may be required by the streaming viewer. (e.g. an mp4 file may need a prefix of "mp4:") (optional)
				<pre>{
				"signedU
				""
				}</pre>
				signedUrl - The URL to b
				the requested content

set-storage-policy	Set Storage Policy	Sets the S3 bucket lifecycle policies associated with a given space. This task is restricted to DuraCloud service administrators.	<pre>{
			"spaceId" : "",
			"storageClass" :
			"",
			"daysToTransition"
			: 0
			}</pre>
			spaceId - Name of the space for which the storage policy should be set
			
			storageClass - One of "STANDARD_IA", "REDUCED_REDUNDANCY", or "GLACIER"
			
			daysToTransition - Number of days content should remain at standard storage before being transitioned to the new storage class
				<pre>{
				"result"
				}</pre>
				result - Text indicating the

noop	Test task	Provides a simple way to test the calling of tasks	None	"Success"

Amazon Glacier Storage Provider

taskName	Name	Description	Request Body	Response Body
restore-content	Restore Content task	Provides the capability to request that specific content items stored in Glacier be retrieved. Content items which are retrieved are made available 3-5 hours after this request is made, and remains available for 2 weeks.	Name of the space and the content item in the form: spaceID/contentID	Text indicating that a restore action has been initiated (or that a restore is already in progress, in the case of duplicate requests.)

Snapshot Storage Provider

taskName	Name	Description	Request Body
create-snapshot	Create Snapshot task	Creates a snapshot by collecting details of the snapshot and passing the request down to a bridge application which makes a copy of the contents of the space.	`{ "spaceId" : "", "description" : "", "userEmail" : "" }`
get-snapshot	Get Snapshot task	Retrieves the status and details of a snapshot action	`{ "snapshotId" : "" }`
cleanup-snapshot	Clean Up Snapshot task	Handles the removal of content items in a space after a snapshot has taken place	`{ "spaceId" : "" }`
complete-snapshot	Complete Snapshot task	Completes the snapshot process	`{ "spaceId" : "" }`

complete-cancel-snapshot	Complete the cancellation of a snapshot	Handles the removal of any space properties, .collection-snapshot.properties file, and snapshot related user permissions. It should be called by the bridge after it has finished its cancellation process.	<pre>{
			  "spaceId" : ""
			}</pre>

restart-snapshot	Restart Snapshot task	Restarts the snapshot process if a failure occurred while transferring from DuraCloud to the bridge.	<pre>{
			  "snapshotId" : ""
			}</pre>

get-snapshots	Get List of Snapshots task	Retrieves a listing of all snapshots which have been created	None

get-snapshot-contents	Get List of Snapshot Contents task	Retrieves a listing of the contents of a particular snapshot	<pre>{
			  "snapshotId" : "",
			  "pageNumber" : 0,
			  "pageSize" : 1000,
			  "prefix" : ""
			}</pre>

get-snapshot-history	Get Snapshot History task	Retrieves a listing of events which have occurred in the history of a particular snapshot	`{ "snapshotId" : "", "pageNumber" : 0, "pageSize" : 0 }`
restore-snapshot	Restore Snapshot task	Requests that a snapshot be restored to a DuraCloud space	`{ "snapshotId" : "", "userEmail" : "" }`
complete-restore	Complete Restore task	Completes the restoration action by setting up an expiration policy for restored content	`{ "spaceId" : "", "daysToExpire" : 1 }`
get-restore	Get Snapshot Restore task	Retrieves the status and details of a restore action. Note that you must specify either the snapshotId or the restoreId, but not both. Specifying the snapshotId will return the most recent restoration matching that snapshotId. Specifying the restoreId you will get back the restoration matching that ID (as you would expect).	`{ "snapshotId" : "", "restoreId" : "" }`

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>
  <!--optional-->
  <millDbEnabled>[true or false]</millDbEnabled>
</duradminConfig>
```

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

Is Initialized

- Purpose: Performs a check to determine if the DurAdmin application has been initialized
- Request: `GET https://host:port/duradmin/init`
- Response Code: 200 (if the application has been initialized), 503 (if the application has NOT been initialized)
- Response Body: Text indicating whether initialization has occurred.

DuraBoss

Purpose: DuraBoss provides administrative control over a variety of activities that run over the storage managed by DuraCloud.

- Reporter - generates reports relating to the status of the storage and services within DuraCloud

Initialization REST Methods

Initialize Application

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

```
<durabossConfig>
  <reporterEnabled>[true|false]</reporterEnabled>
  <durastoreHost>[host]</durastoreHost>
  <durastorePort>[port]</durastorePort>
  <durastoreContext>durastore</durastoreContext>
  <notificationConfig>
    <type>EMAIL</type>
    <username>[username for notification system]</username>
    <password>[password for notification system]</password>
    <originator>[from email address]</originator>
    <admin>[administrator email address]</admin>
  </notificationConfig>
</durabossConfig>
```

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

Is Initialized

- Purpose: Performs a check to determine if the DuraBoss application has been initialized
- Request: `GET https://host:port/duraboss/init`
- Response Code: 200 (if the application has been initialized), 503 (if the application has NOT been initialized)
- Response Body: Text indicating whether initialization has occurred.

Reporter: Storage Report REST Methods

Get Latest Storage Report

- Purpose: Provides the most current storage report in XML format
- Request: `GET https://host:port/duraboss/report/storage`
- 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/duraboss/report/storage/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/duraboss/report/storage/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/duraboss/report/storage/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/duraboss/report/storage`
- 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/duraboss/report/storage`
- 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/duraboss/report/storage/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/duraboss/report/storage/schedule`
- Response Code: 200 (on success)
- Response Body: "Storage Reports schedule cancelled"

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 Properties

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

Get Space Properties

```
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 Properties

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

Get Content Properties

```
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
```

DuraBoss Report API

Get Latest Storage Report

```
curl -u ${user}:${pword} https://${host}/duraboss/report/storage
```

Get Storage Report List

```
curl -u ${user}:${pword} https://${host}/duraboss/report/storage/list
```

Get Storage Report Info

```
curl -u ${user}:${pword} https://${host}/duraboss/report/storage/info
```

Start Storage Report

```
curl -u ${user}:${pword} -X POST https://${host}/duraboss/report/storage
```

Stop Storage Report

```
curl -u ${user}:${pword} -X DELETE https://${host}/duraboss/report/storage
```

Schedule Storage Report (to begin Jan 1, 2020 at 01:01:01 and repeat every 10 min)

```
curl -u ${user}:${pword} -X POST
https://${host}/duraboss/report/storage/schedule?startTime=1577840461000\&
frequency=600000
```

Cancel Storage Report Schedule

```
curl -u ${user}:${pword} -X DELETE
https://${host}/duraboss/report/storage/schedule
```

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. To get started with the Sync Tool, read on, or watch videos of the process here.

Download and Install

Download installers for Mac OSX, Windows, or Linux from the DuraCloud Downloads page.

The Sync Tool requires that Java version 7 or above (Java 8 is preferred) be installed on your system in order to run. The installers for your operating system will check to make sure that you have the correct version of Java and will prompt you to download and install Java if necessary. If you would like to update Java directly, it can be downloaded from here.

If you are using a Mac and have questions about Java 7, you will likely find answers here. Due to issues which occasionally arise when attempting to install Java 7 on the Mac platform, an installer is provided which bundles Java 7, removing the need to perform a local installation.

If you cannot use Java 7 on your platform now, or would simply prefer to use Java 6, you have the option of using an older version of either the Sync Tool (which is available here) or the Upload Tool (which is available here). This version of the Sync Tool only includes a command line interface. The Upload Tool provides a graphical interface (a GUI), but is not as full featured as the Sync Tool.

Graphical Interface

The Sync Tool defaults to a graphical user interface.

- The Sync Tool provides a web-browser-based application user interface which begins with a configuration wizard, then provides a dashboard display showing the current status of the sync process. This interface is the default and is started by running selecting any of the shortcuts created by the installer.
- Once running, this interface will be continually available at this address: http://localhost:8888/sync.

Operational Notes

- Closing the browser window will not stop the Sync Tool. It will continue to run and transfer files.
- Getting back to the Sync Tool
    - Once the Sync Tool is started, it will continue to run in the background, even if you close your browser. You can also get back to the UI by pointing your browser to: http://localhost:8888/sync (hint: bookmark this page).
    - Selecting any of the shortcuts created by the installer will bring up the Sync Tool UI.
- Stopping the Sync Tool

        - Within the UI there are options to stop and start the sync. This will allow you to stop syncing for a time, and start it up again later.
- Stopping the Sync Tool process
    - If you would like to completely shut down the Sync Tool process, such that the UI is no longer available:
        - On Windows: Look for a DuraCloud Sync icon in the task tray, right click on it, select Exit
        - On Mac: Look for a DuraCloud Sync icon in the menu bar, right click on it, select Exit
        - On Ubuntu: Look for a DuraCloud Sync icon in the task bar, right click on it, select Exit
- Work Directory
    - The work directory is named duracloud-sync-work, and can be found under your home directory (C:\Users\[username] on Windows, /Users/[username] on Mac, /home/[username] on Linux)
    - In the work directory you will find:
        - A configuration file which includes the data you entered when configuring the tool
        - A logs directory with log files containing runtime status information of the Sync Tool. These can be helpful when diagnosing a problem the tool may have had.
- Jump Start
    - The Jump Start option available in the SyncTool is designed to streamline the transfer of new file sets to DuraCloud. This is accomplished by removing the checks that the SyncTool traditionally performs before uploading a file. These checks normally try to determine if a file already exists in DuraCloud. With the Jump Start option enabled, the SyncTool assumes that all files are new and need to be moved to DuraCloud. This is option is ideal for the initial data transfer into DuraCloud, when all selected data needs to be transferred. The Jump Start option should be turned off when running the SyncTool over a data set that is already in DuraCloud (in order to discover and transfer any new files), so that unnecessary content transfers can be avoided.
- Transfer Rate Optimization
    - When performing a transfer of files to DuraCloud, the goal is often to get those files moved as quickly as possible. To assist with this the Sync Tool allows you to adjust the number of simultaneous transfers (a.k.a "threads") on the Configuration tab. One caveat here is that the higher the number of threads, the more system resources will be consumed. Additionally as the number of simultaneous transfers increases, more network bandwidth will be consumed. So what is the best number of threads? It depends on the characteristics of your machine and the network to which it is attached. We've added **a handy new feature** that will automatically determine the optimal number of threads for your system. In the "Transfer Rate" section of the configuration page you'll notice an "Optimize Automatically" button. Click it to automatically discover and set the optimal number of threads.
    - Note that "optimal", in this context, means the number of threads that will allow content to be transferred as quickly as possible.
    - The determination of "optimal" thread count is based on testing actual timed transfers, which is why the test may take a while to run when resources such as upload bandwidth or CPU or memory capacity are constrained. This also means that the optimal thread count given will reflect the capability of the machine while the test is running. If other tasks on the machine are consuming significant system resources, this will affect the results of the test.
    - If the machine being used for the transfer of content is not primarily dedicated to this one task (at least while the SyncTool is running), then you may want to set the thread count lower than the determined "optimal" setting. This will, of course, reduce the transfer rate, but would allow the machine to have capacity for other activities.
    - You can use the SyncOptimize Tool to perform these tests if you would prefer to run them on the command line, have more control over the parameters used, and see more details about the testing process.
- Destination Prefix
    - Using the prefix option, the content IDs that are created for the files being moved to DuraCloud by the SyncTool can be made to begin with a consistent text value. There are several reasons this might be useful, such as to include the name of a top-level directory in the path, or to be able to run the Sync from a new sub-directory, but still maintain the full path included on all existing stored content. Suppose the path to a local file (found within the watch directory) is "dir1/file.txt" and you would like the resulting content stored in DuraCloud to be 'a/b/c/dir1/file.txt. To achieve that result, the destination prefix of "a/b/c/" would need to be set.

        > Adding or changing a prefix for content that has already been transferred to DuraCloud will result in those files being duplicated in DuraCloud storage with the new prefix. Removing the duplicate files can be done by using the "sync deletes" option, but this will cause all content in the destination space which does not include the prefix to be deleted (along with any content that is not found in the local watch directories.) Be cautious when using this feature if you have already uploaded content to your DuraCloud space.

        > If you use a prefix to include a file path (such as a top level directory name), remember to include the "/" character at the end of your prefix. For example, using the prefix "dir1/" with file "file.txt", your final content ID will be "dir1/file.txt". If you were to forget the slash, your prefix would be "dir1", which would lead to a content ID of "dir1file.txt", which is likely not what you want.

- Run Modes
    - You may run your synchronization operations in one of two modes: continuous or single pass. In continuous mode, DuraCloudSync will continue indefinitely to watch for additions, updates, and deletions to the file system after adding all the files in your watched directories when the sync starts. In the single pass mode, the application will not continue to watch for changes after making the initial pass of your configured directories and/or files.

Command Line Interface

The Sync Tool provides a command line interface which can be executed directly, used in scripts, or used for scheduling sync activities (such as

within a cron job.) The command line interface provides access to all feature of the Sync Tool, some of which are not available (yet) in the graphical interface.

- More information about using the command line interface can be found here.

Metadata

As the Sync Tool transfers files to DuraCloud, it will attempt to capture certain types of metadata about each file, and include that information as part of the content item added to DuraCloud. The list below describes the metadata that is captured automatically. You have the option to add, update, or delete the properties of each file after it has been transferred to DuraCloud.

- Mime Type
  - The content type of the file.
  - As the Sync Tool transfers your files to DuraCloud, it attempts to determine the mime type of each file based on the file's extension. If it cannot determine a mime type for a given file, that file's type is set to "application/octet-stream", which is a generic mime type for binary data. Select the "Edit" button on the DuraCloud web interface to change a file's mime type.
  - If you find that files with certain extensions are not being mapped as you would prefer, you can always change the value on uploaded files from within DuraCloud. If you would like to make sure that files with a given extension are given your preferred mime type during upload, you simply need to update the mapping file. The mapping of file extension to mime type is determined by a file included in your Java installation called content-types.properties. This file is usually located in the "lib" folder under your Java runtime installation directory. After making a copy of the original file as a backup, simply update it following the formatting conventions used throughout the file to include the mappings you prefer, then save the file. After making changes, you will need to re-start the Sync Tool to ensure that the changes are picked up properly.
- Space
  - The space in which a content item is stored. This field cannot be edited.
- Size
  - The size of a content item. This field cannot be edited.
- Modified
  - The date on which the file was added to DuraCloud. This value is updated when a file is added or updated.
- Checksum
  - The MD5 checksum of the file. This field cannot be edited.
- Creator
  - The creator is the DuraCloud user who transferred the file into DuraCloud storage.
- Content file path
  - The full path of the file in its original storage location
- Content file created
  - The date when the file was created, as determined by the originating file system. This information may not be available from all file systems.
- Content file modified
  - The date when the file was last modified, as determined by the originating file system. This information may not be available from all file systems.
- Content file last accessed
  - The date when the file was last accessed, as determined by the originating file system. This information may not be available from all file systems.

DuraCloud SyncOptimize Tool

Introduction

The SyncOptimize is a tool that will help you determine the best number of threads to configure for optimal throughput given the resources available on your computer and network. Before running the Sync Tool you may want to run SyncOptimize first. Once it has completed running diagnostic tests, it will return a value that you can plug into the -t parameter of the Sync Tool. If you're using Sync Tool's graphical interface, you can invoke SyncOptimize directly without having to run the command line tool. You can fire up the optimizer either by clicking "Optimize" on the final step of the setup wizard or you can access it on the Configuration tab. For more information on running the Sync Tool, go to the DuraCloud Sync Tool page.

Download

Download the sync optimize tool from the Downloads page.

Using the Sync Optimize Tool

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

- To display the help for the Sync Optimize Tool, run

```
java -jar syncoptimize-{version}-driver.jar --help
```

- When running the Sync Optmize 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
-s	--space	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	No	The password necessary to perform writes to DuraStore. If not specified the retrieval tool will first check to see if an environment variable named "DURACLOUD_PASSWORD" exists, if it does exist the retrieval tool will use its value as the password, otherwise you will be prompted to enter the password.	Not set
-m	--size-files	Yes	No	The size of files to transfer on each test run, in MB	5
-n	--num-files	Yes	No	The number of files to transfer on each test run	10

- Examples:
  - Run the Sync Optimize Tool with default size and number of test files

    ```
    java -jar syncoptimize-{version}-driver.jar -h test.duracloud.org -s test-space -u myname -p mypassword
    ```

  - Run the Sync Optimize Tool with 50MB test files (perhaps because the average file size in the data sets that will be transferred is 50MB)

    ```
    java -jar syncoptimize-{version}-driver.jar -h test.duracloud.org -s test-space -u myname -p mypassword -m 50
    ```

DuraCloud Sync Tool - Command Line

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 from the Downloads page.

Getting Started

The Sync Tool can be installed using one of the installers on the downloads page linked above. Once installed, the Sync Tool will default to running in GUI mode. To run in command line mode, open a terminal window (or command prompt) and navigate to the Sync Tool installation directory. Once there, execute the Sync Tool JAR file using: "java -jar duracloudsync.jar --help". This will print the usage information for the tool.

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 and use the same options, 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

- Running
    - To ensure that the command line interface is selected, at least one command line option must be included when executing the Sync Tool on the command line. To see the help text, simply include a "--help" parameter. Running with no parameters will start the Sync Tool in GUI mode.
- Jump Start
    - The Jump Start option available in the SyncTool is designed to streamline the transfer of new file sets to DuraCloud. This is accomplished by removing the checks that the SyncTool traditionally performs before uploading a file. These checks normally try to determine if a file already exists in DuraCloud. With the Jump Start option enabled, the SyncTool assumes that all files are new and need to be moved to DuraCloud. This is option is ideal for the initial data transfer into DuraCloud, when all selected data needs to be transferred. The Jump Start option should be turned off when running the SyncTool over a data set that is already in DuraCloud (in order to discover and transfer any new files), so that unnecessary content transfers can be avoided.
- 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.
- Getting a clean start
    - If you specifically do not want to restart from a previous run, and would like to ensure that the sync tool considers every file in all directories specified, you can use the -l (or --clean-start) command line option to indicate this desire.
    - A clean start will also occur by default whenever the host, destination space, destination store, or the list of content directories changes from one run of the tool to the next.
- 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 default
    - As of DuraCloud version 2.3.0, the work directory parameter is not required. If not specified, the work directory will be named "duracloud-sync-work", and will be placed under the user's home directory
- 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.
- Time Stamps
    - As of DuraCloud version 2.3.0, the Sync Tool will collect time stamp information for each transferred file from the file system and

store this information as properties on the content item in DuraCloud
    - Note that the time stamps collected may vary somewhat based on the operating system and file system on which the content is stored
- Destination Prefix
    - Using the prefix option, the content IDs that are created for the files being moved to DuraCloud by the SyncTool can be made to begin with a consistent text value. There are several reasons this might be useful, such as to include the name of a top-level directory in the path, or to be able to run the Sync from a new sub-directory, but still maintain the full path included on all existing stored content. Suppose the path to a local file (found within the watch directory) is "dir1/file.txt" and you would like the resulting content stored in DuraCloud to be 'a/b/c/dir1/file.txt. To achieve that result, the destination prefix of "a/b/c/" would need to be set.

> Adding or changing a prefix for content that has already been transferred to DuraCloud will result in those files being duplicated in DuraCloud storage. Removing the duplicate files can be done by using the "sync deletes" option, but this will cause all content in the destination space which does not include the prefix to be deleted (along with any content that is not found in the local watch directories.) Be cautious when using this feature if you have already uploaded content to your DuraCloud space.

> If you use a prefix to include a file path (such as a top level directory name), remember to include the "/" character at the end of your prefix. For example, using the prefix "dir1/" with file "file.txt", your final content ID will be "dir1/file.txt". If you were to forget the slash, your prefix would be "dir1", which would lead to a content ID of "dir1file.txt", which is likely not what you want.

- Optimizing Transfer Rate
    - You can potentially increase the transfer rate of your content by increasing the thread count. The thread count can be set using the -t option. To help you determine the optimal thread count in order to maximize throughput, we've added a new diagnostic tool. Please see DuraCloud SyncOptimize Tool for more information.

Prerequisites

> As of DuraCloud version 2.2.0, the Sync Tool requires Java 7 to run. As of version 3.3.0, DuraCloud is primarily tested using Java 8, and this is the recommended Java version for building and running DuraCloud tools. The latest version of Java can be downloaded from here.

- You must have Java version 7 or above installed on your local system (Java 8 is preferred). If Java is not installed, or if a previous version is installed, you will need to download and install Java. 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 1.7.0 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.

Using 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 duracloudsync-{version}.jar --help
```

- 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	No	The password necessary to perform writes to DuraStore. If not specified the sync tool will first check to see if an environment variable named "DURACLOUD_PASSWORD" exists, if it does exist the sync tool will use its value as the password, otherwise you will be prompted to enter the password.	Not set
-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.	
-j	--jump-start	No	No	This option indicates that the sync tool should not attempt to check if content to be synchronized is already in DuraCloud, but should instead transfer all content. This option is best used for new data sets.	Not set
-a	--prefix	Yes	No	A prefix to be added to the content IDs of all files in the content directories as they are added to DuraCloud. The same prefix applies to all files in all content directories. For example, if a content directory is C:/users/bob/pictures with one file in it, C:/users/bob/pictures/001.jpg, and the prefix value is "bobs-pictures/", the file would be given a DuraCloud content ID of bobs-pictures/001.jpg. Note that the name of the content directory is not included in the path, so if you would like for it to appear as part of the content ID, you will need to include it in the prefix. Also note that the prefix does not need to be a directory name, it can be any value. If, however, you would like for it to appear as a directory path, do not forget to end the prefix with a "/" character.	Not set
-w	--work-dir	Yes	No	The state of the sync tool is persisted to this directory. If not specified, this value will default to a directory named duracloud-sync-work in the user's home directory.	duracloud-sync-work
-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
-n	--rename-updates <suffix>	No	No	<suffix> indicates that updates should be synced to the cloud and renamed. Specify an optional suffix to override default (".orig"). To prevent updates altogether, see option -o. (Note that this option cannot be used together with either the -o or the -d options.)	
-o	--no-update	No	No	Indicate that changed files should not be updated. In order to perform updates without overwriting, see option -n.	
-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
-l	--clean-start	No	No	Indicates that the sync tool should perform a clean start, ensuring that all files in all content directories are checked against DuraCloud, even if those files have not changed locally since the last run of the sync tool	Not set
-e	--exclude	Yes	No	The full path to a file which specifies a set of exclusion rules. The purpose of the exclusion rules is to indicate that certain files or directories should not be transferred to DuraCloud. The rules must be listed one per line in the file. The rules will match only on the name of a file or directory, not an entire path, so path separators should not be included in rules. Rules are not case sensitive (so a rule "test.log" will match a file "test.LOG"). The rules may include wildcard characters ? and *. The ? matches a single character, while * matches 0 or more characters. Examples of valid rules: test.txt : Will match a file named "test.txt" test : Will match a file or directory named "test" test.* : Will match files like "test.jpg", "test.txt", "test.doc", etc *.log : Will match files named "test.log" or "daily-01-01-2050.log" as well as a directory named ".log" backup-19?? : Will match a directory named "backup-1999" but not "backup-190000" or "backup-2000"	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)

Examples of commands to use the command line SyncTool:

- Command to sync the contents of a single local content directory to DuraCloud.

```
java -jar duracloudsync-{version}.jar -c C:\files\important -h test.duracloud.org
-s important-dir-backup -u myname -p mypassword
```

- Command to sync the contents of multiple local content directories to DuraCloud.

```
java -jar duracloudsync-{version}.jar -c C:\files\important
C:\Users\me\Documents\important -h test.duracloud.org -s important-dir-backup -u
myname -p 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. These commands are not available when running in exit-on-completion mode.

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

Running the Sync Tool in a server shell environment

As noted above, the Sync Tool can be run in one of two modes, one which allows it to run continually, and the other which allows it to exit once it completes transferring all current files. The mode you choose will determine the way in which you deploy the Sync Tool on a server. The following examples assume the use of the bash shell.

To start the Sync Tool in continually running mode, you would use a command like this:

```
nohup java -jar duracloudsync-{version}.jar {parameters} > ~/synctool-output.log 2>&1
&
```

In this case, the & at the end of the command instructs the command to run in the background, and the "nohup" at the beginning tells the command to continue running even when the terminal being used is closed or when you disconnect from the server machine. The output of the Sync Tool would be placed in a file called "synctool-output.txt" in the user's home directory.
In order for the Sync Tool to be run on startup when the server machine boots, additional settings will need to be added which depend on the operating system being used. In Ubuntu, for example, an Upstart script would be used for this purpose.
Running the Sync Tool in exit on completion mode works best when the tool is run on a scheduled basis. A popular choice for handling this type of task is the cron utility. To run daily using cron a script should be placed in /etc/cron.daily. The script would look something like:

```
#!/bin/bash

java -jar duracloudsync-{version}.jar -x [parameters] >> ~/synctool-output.log 2>&1
```

The -x parameter is included here to ensure the Sync Tool exists after completing its run.

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.
- A file containing the list of content files within a space can be created using the "list-only" option (-l) instead of retrieving the actual content files themselves. The format of this text file is one content file name per line. This can be useful for many things.
- Specific content files can be retrieved from a space using the "list-file" option (-f) instead of retrieving all content files from a space. This can be useful by saving lots of time and bandwidth usage. One way to do this would be to first run a retrieval-tool command to create a file containing all content file names in a space using the "list-only" option. Then editing the text file containing the list of content names so it only contains a list of the desired content names and then use this file with the "list-file" option.

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. An output file is created for each run of the Retrieval Tool which stores a listing of the files that were downloaded.

Prerequisites

> As of DuraCloud version 2.2.0, the Retrieval Tool requires Java 7 to run. The latest version of Java can be downloaded from here.

- You must have Java version 7 or above installed on your local system. If Java is not installed, or if a previous version is installed, you will need to download and install Java 7. 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 1.7.0 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.

Using 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	No	The password necessary to perform writes to DuraStore. If not specified the retrieval tool will first check to see if an environment variable named "DURACLOUD_PASSWORD" exists, if it does exist the retrieval tool will use its value as the password, otherwise you will be prompted to enter the password.	Not set
-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	No	Logs and output files will be stored in the work directory. If not specified, this value will default to a directory named duracloud-retrieval-work in the user's home directory.	duracloud-retrieval-work
-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
-d	--disable-timestamps	No	No	Indicates that timestamp information found as content item properties in DuraCloud should not be applied to local files as they are retrieved.	Not set
-l	--list-only	No	No	Indicates that the retrieval tool should create a file listing the contents of the specified space rather than downloading the actual content files. The list file will be placed in the specified content directory. One list file will be created for each specified space.	Not set
-f	--list-file	Yes	No	Retrieve specific contents using content IDs in the specified file. The specified file should contain one content ID per line. This option can only operate on one space at a time.	Not set

Examples of running the retrieval tool:

- Retrieve all the files stored within the 2 specified spaces and place them in the specified local content directory under sub-directories matching the specified 2 space names.

```
java -jar retrievaltool-{version}-driver.jar -c content -h test.duracloud.org -u
myname -p mypassword -s space1 space2 -o
```

- Retrieve all the files stored within all spaces and place them in the specified local content directory under sub-directories matching the space names.

```
java -jar retrievaltool-{version}-driver.jar -c content -h test.duracloud.org -u
myname -p mypassword -a
```

- Create a file containing the list of content IDs for the specified spaces using hidden password option (-p command line option not specified, will be prompted for password). This example would not actually retrieve the content files, rather it creates a list of content files in the specified space. Each specified space will have its own content list file created in the specified local content directory. The naming convention of each list file created will be: "<space_id>-content-listing-<storage_provider>.txt"

```
java -jar retrievaltool-{version}-driver.jar -h <host> -u <user> -c <content_dir>
-s <list of space IDs separated by a space> -l
```

- Retrieve only the specified contents by using the list-file option (-f). The -f option can only operate on one space. This command would result in having all the content files listed in the specified file of content IDs placed in the specified local content directory.

```
java -jar retrievaltool-{version}-driver.jar -h <host> -u <user> -c <content_dir>
-f <path_to_file_of_specified_content_IDs> -s <single space ID>
```

DuraCloud Features

Web Interface Features

- Access to all storage system capabilities, including space and content creation, updates, and deletes
- Access to graphical depictions of the information contained in the storage reports
- Bulk deletion of spaces and content items
- User account administration

API Features

- Storage system REST API
  - Space (top level folder) listing
  - Space creation
  - Space deletion
  - Space access definition
  - Content (file) listing
  - Content storage
  - Content deletion
  - Content properties and tagging
  - Retrieval of audit logs
  - Retrieval of space manifest

  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

Core Services

- **Media Streaming:**
  - Provides streaming capabilities for video and audio files.
- **Bit Integrity Checking:**
  - Verifies that content stored in DuraCloud maintains bit-level integrity over time by calculating a checksum value upon content storage and the at frequent intervals to ensure no changes have occurred

- **Duplication:**
  - Keeps content in multiple storage providers consistent.

Tools

- **Sync Tool:**
  - Provides a utility with both graphical and command line interfaces 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. Files larger than a configurable threshold will be "chunked" (split into multiple files) prior to transfer.
- **Stitcher Tool:**
  - Provides a command line utility for retrieving single "chunked" files from DuraCloud. As the file is retrieved, it is "stitched" (combined back into the original file).

DuraCloud Storage

Introduction

DuraCloud provides a mediation layer over cloud storage systems. The architecture of DuraCloud is such that several of these storage systems, which DuraCloud calls "storage providers", can be configured and put into use at one time. DuraCloud expects that there will be one primary storage provider, then permits any number of secondary providers to be configured. The primary provider is the location in which content is stored immediately upon user transfer; content is then automatically copied to all configured secondary providers. The lists below indicate the primary and secondary storage providers currently available through the production DuraCloud hosted service, as well as the storage providers which are included in the DuraCloud open source code base, but are not currently considered production level quality.

Current DuraCloud Storage Providers

Primary storage providers

Storage providers which are used as a primary storage location within DuraCloud. A primary provider can be used alone or with any number of secondary providers.

- Amazon S3

Secondary storage providers

Storage providers which are available as a secondary storage location in DuraCloud. Content in the primary storage provider is duplicated into all configured secondary providers.

- SDSC Cloud Storage
- Rackspace Cloud Files
- Amazon Glacier
- Snapshot - this storage provider is used as part of the DPN node integration which is run in coordination with Chronopolis, however it can be used to produce snapshots of space content sets for anyone running DuraCloud software.

Development storage providers

Storage providers which are not made available in production DuraCloud environments because their current level of development and/or testing is not yet sufficient. Providers on this list may be scheduled to move into production status or may be held back due to technical or business reasons. These providers are available as part of the open source DuraCloud codebase, but it is not recommended that they be used in a production environment without further development.

- IRODS

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 four DuraCloud web applications (/duradmin, /durastore, /duraservice, and /duraboss) 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]

  ###
  # try to require https for 'durastore' & 'duraboss' for
  # external requests
  ###

  RewriteCond %{REQUEST_URI} ^(/durastore|/duraboss)
  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:

Initialization REST Methods - Common across all applications

Action	Role
Is Initialized	ROLE_ANONYMOUS
Initialize	ROLE_ROOT
Initialize Security Users	ROLE_ROOT

DuraStore REST Methods

Action	Role
Get Stores	ROLE_USER
Get Spaces	ROLE_ANONYMOUS if space ACL allows public read, else ROLE_USER
Get Space	ROLE_ANONYMOUS if space ACL allows public read, else ROLE_USER
Get Space Properties	ROLE_ANONYMOUS if space ACL allows public read, else ROLE_USER
Get Space ACLs	ROLE_ANONYMOUS if space ACL allows public read, else ROLE_USER
Create Space	ROLE_ADMIN
Set Space Properties	ROLE_USER
Set Space ACLs	ROLE_ADMIN
Delete Space	ROLE_ADMIN
Get Content	ROLE_ANONYMOUS if space ACL allows public read, else ROLE_USER
Get Content Properties	ROLE_ANONYMOUS if space ACL allows public read, else ROLE_USER
Store Content	ROLE_USER
Copy Content	ROLE_USER
Set Content Properties	ROLE_USER
Delete Content	ROLE_USER
Get Tasks	ROLE_ADMIN
Perform Task	ROLE_ADMIN
Perform Task (restore-content)	ROLE_ROOT

DuraBoss REST Methods

Action	Role
Get Latest Storage Report	ROLE_ADMIN
Get Storage Report List	ROLE_ADMIN
Get Storage Report	ROLE_ADMIN
Get Storage Report Info	ROLE_ADMIN
Start Storage Report	ROLE_ROOT
Cancel Storage Report	ROLE_ROOT
Schedule Storage Report	ROLE_ROOT
Cancel Storage Report Schedule	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
    - administrator of DuraCloud-account
4. ROLE_ROOT
    - DuraSpace personnel

User Management and Access Control

- Users are managed via the DuraCloud Management Console. In the Management Console, an account administrator has the ability to:
    1. Add and remove users to the DuraCloud account
    2. Create Groups and add users to groups in order to simplify access control
- Access Control is managed at the space level
    - Within DuraCloud (via the UI or the REST API), an account administrator has the ability to define which users and groups have access to a space, as well as the type of access (read or write) that is available.

DuraCloud Administration

- Naming restrictions
- Access Control Lists (ACLs)

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 42 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
        - init
        - 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
        - is limited to 1024 bytes (byte count is checked after URL and UTF-8 encoding)

Access Control Lists (ACLs)

Access control in DuraCloud is set at the space level. Users and groups can be provided read and write access to a space.

1. Users and Groups
    a. Access is granted to *users*, *groups*, or combinations thereof
    b. *Users* are those with credentials to access an account
    c. *Groups* are collections of users that are created in the Management Console
2. Rights
    a. When assigning a space ACL, *users* and/or *groupsv* are granted one of two rights
        i. *READ* allows reading any content within that space
        ii. *WRITE* allows reading, adding, and modifying any content within that space
3. Public (anonymous) Access
    a. There is a special group named 'public' that can only be granted *READ* access to a space
    b. If the 'public' group has *READ* access, then unauthenticated (anonymous) reads of content are permitted on that space
4. Use
    a. REST API can be used to programmatically create, update, and delete space ACLs
        - Get Space ACLs
        - Set Space ACLs
    b. DurAdmin provides authorized users to update space ACLs in the web interface

DuraCloud Java Clients

Download

Download all of the clients listed below 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.

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 as part of an application called DuraBoss. DuraBoss is installed and running on your DuraCloud instance and the reporting capabilities can be accessed via a REST interface. In order to aid Java developers in communicating with the reporting features of DuraBoss, a Java client, called ReportClient was written.

Using the clients

To use any of the above clients, you will need to include their dependencies. This is best done using Maven, as follows:

StoreClient

```
<dependency>
  <groupId>org.duracloud</groupId>
  <artifactId>storeclient</artifactId>
  <version>{duracloud-version}</version>
</dependency>
```

ReportClient

```
<dependency>
  <groupId>org.duracloud</groupId>
  <artifactId>reportclient</artifactId>
  <version>{duracloud-version}</version>
</dependency>
```

Alternatively, all of the jars included in the libs directory of the download package can be added to your local java classpath. You will then be able to write code using the provided Javadocs to interact with the client.

Example code

StoreClient

An example Maven-based project which provides examples for using the storeclient can be found on Github here: https://github.com/duracloud/rest-client-example

ReportClient

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 the download package under /resources) includes a simple main class which performs a subset of 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 Media Streaming

Introduction

The Media Streaming capabilities provided by DuraCloud allow video and audio files to be streamed over RTMP. This feature in DuraCloud takes advantage of Amazon Cloudfront streaming, so files to be streamed must be within spaces on an Amazon provider. 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.

Open and Secure Streaming

DuraCloud supports two different types of streaming, open and secure. Open streaming allows anyone with access to the URL for a streamed content item to stream the content. This works well for open access content which is intended to be shared and accessed widely. Secure streaming requires that a request be made to DuraCloud to retrieve a signed URL. That signed URL can then be used to stream the file. The request to retrieve a signed URL can specify how long the content will be available to stream (the default is 8 hours) as well as the IP address or IP address range where the streaming is allowed to take place. The purpose of secure streaming is to restrict the use of the stream. This is ideal for scenarios where streamed content is not free to use or must only be provided to a limited audience. Note that both types of streaming, open and secure, use the RTMP protocol, which protects the source file from being downloaded. The RTMP protocol requires that a flash-based streaming media player be used to play the streamed content.

Using Media Streaming

Follow these steps to stream media files with DuraCloud

1. Create a space in DuraCloud which you will use to host streamed files
2. Transfer media files to the space. Be sure that the file are using supported formats (see the link above).
3. Enable streaming
    a. To enable open streaming: Select the space in the DuraCloud interface and click the "ON" button next to "Streaming:" in the top row of buttons.
    b. To enable secure streaming: Perform a POST HTTP call to the URL https://{institution}.duracloud.org/durastore/task/enable-streaming. The body of the POST request should include this JSON document: {"spaceId" : "","secure" : "true"}. Fill in the ID of the space to stream. Note that this call is using the DuraCloud REST API.
4. Wait up to 15 minutes. If this is the first time the space has been streamed, it can take up to 15 minutes for the files to be available on the Amazon edge servers.
5. Stream a file
    a. When using open streaming:
        i. Select a media file in the space. A video player will appear in the Content Detail pane. Verify that you are able to play the streamed file.
        ii. Look in the space properties for the RTMP streaming address. This is the path you will use for streaming files. Alternatively, you can perform a get-url task call through the DuraCloud REST API to retrieve the streaming URL for each content item to be streamed. These URLs are predictable and do not expire.
    b. When using secure streaming:
        i. Spaces using secure streaming do not provide playback via the DuraCloud UI. You will need to perform a "get-signed-url" call to retrieve a signed URL for each content item to be streamed, and stream the file through an RTMP compatible player. More details about the get-signed-url call can be found in the Amazon S3 Storage Provider tasks section of the DuraCloud REST API.
6. Set up your website or application to provide access to the streamed files. Some example files to get you started are listed below.

> The Flash Media Server used by Amazon Cloudfront and media players like JWPlayer and Flowplayer require certain specific conventions for requesting streamed files. There are two primary variables, one being a prefix which may need to preceed the file name (example prefix values are "mp3:" and "mp4:"). The other variable is whether a file extension is allowed on the file name. Getting these combinations right is particularly important when using secure streaming, as the player cannot request the file with alternative file names to match its preferences. Not all file types use the same combination of prefix and file extension settings. For example, it is common for MP4 files to require a prefix and extension (example file name: "mp4:videofile.mp4") while MP3 files require a prefix but no extension (example file name: "mp3:audiofile").
>
> The prefix value, when needed, should be added to the stream path by using the "resourcePrefix" parameter on the get-url or get-signed-url call made through the DuraCloud REST API.
>
> In most cases, the file extension will need to be part of the stored file name. Even if files are named with a file extension (which is typically the case), calls to retrieve a streaming URL can specify the file name with no extension.

Integration Files

The following files are available as a bundle on the downloads page.
They are intended as a starting point for integrating streaming media into your own website.

- player.swf - The flash-based video player JWPlayer
- playlist.xml - An example playlist which would include a list of items in your Source Media Space
- playlistplayer.html - An HTML file which uses JWPlayer to display the items in the playlist
- singleplayer.html - An HTML file which uses JWPlayer to display a single media file
- 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 used to simplify the loading of JWPlayer

All of the above 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.

> If you add files to a space with streaming turned on, those files will automatically be made available for streaming as well.

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.

> In most cases, the Sync Tool is a better choice for transferring files to DuraCloud. The Chunker Tool is made available only for the rare cases when you need very specific control over exactly how files are split into chunks prior to upload.

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.

Prerequisites

> As of DuraCloud version 2.2.0, the Chunker Tool requires Java 7 to run. The latest version of Java can be downloaded from here.

- You must have Java version 7 or above installed on your local system. If Java is not installed, or if a previous version is installed, you will need to download and install Java 7. 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 1.7.0 or above. If running this command generates an error, Java is likely not installed.
- You must have downloaded the Chunker Tool. It is available as a link near the top of this page.

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

DuraCloud Stitcher Tool

Introduction

The Stitcher Tool is a utility which provides a simple way to retrieve "chunked" files from DuraCloud. When files are moved to DuraCloud using either the Chunker Tool or the Sync Tool and they exceed a defined size limit, they are split (chunked) into multiple files for transfer. The Stitcher tool provides the means by which those files can be retrieved and combined to result in the original file. It should be noted that the Stitcher Tool is also embedded into the Retrieval Tool. If you are using Retrieval Tool, stitching will be automatically performed.

Download

Download the Stitcher 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 stitch-{version}-driver.jar
```

  The resulting usage statement (detailed below) should be enough to help you get started.
- When using the Stitcher Tool, you need to know the ID of the manifest which was generated to list all of the chunks of the original file. If the chunking was done by either the Chunker or Sync tool, then the name of the manifest is the name of the original file (prefixed with any enclosing directory names) followed by ".dura-manifest".

Prerequisites

As of DuraCloud version 2.2.0, the Stitcher Tool requires Java 7 to run. The latest version of Java can be downloaded from here.

- You must have Java version 7 or above installed on your local system. If Java is not installed, or if a previous version is installed, you will need to download and install Java 7. 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 1.7.0 or above. If running this command generates an error, Java is likely not installed.
- You must have downloaded the Stitcher Tool. It is available as a link near the top of this page.

Using the Stitcher Tool

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

Short Option	Long Option	Argument Expected	Required	Description	Default Value (if optional)
-m	--manifest-id	Yes	Yes	The ID of the manifest file used to contain the listing of content chunks	
-d	--to-dir	Yes	Yes	Retrieved and stitched content is stored in this local directory	
-s	--space-id	Yes	Yes	The space ID in which content and manifest files reside	
-i	--store-id	Yes	No	The store ID for the DuraCloud storage provider	The default store is used
-h	--host	Yes	Yes	The host address of the DuraCloud instance	
-r	--port	Yes	No	The port of the DuraCloud instance	443
-u	--username	Yes	Yes	The username necessary to perform writes to DuraCloud	
-p	--password	Yes	Yes	The password necessary to perform writes to DuraCloud	

Known Issues

The following issues are known to exist in release 3.1.0 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** - Incorrect redirects from http to https OPEN	https:// when performing requests via the REST API of hosted DuraCloud instances
Administrative files in storage report graphs	The storage report information displays all files in DuraCloud, including all of the administrative files	**DURACLOUD-514** - Filter administrative content from storage reports OPEN	View reports for content spaces directly

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

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"

Deploying DuraCloud from Binaries

If you run into problems trying to install and run DuraCloud, the best place to ask questions is on the DuraCloud Dev mailing list

> As of version 2.2.0, **DuraCloud requires Java 7.** To determine the version of Java installed on your system, open a terminal (command prompt) and run the command "java -version". If the version number printed is less than 1.7, you will need to upgrade your Java version prior to running DuraCloud. As of version 3.3.0, **DuraCloud is primarily tested using Java 8**, and this is the recommended Java version for building and running DuraCloud.

The steps below outline how to start up a DuraCloud instance from the binary distribution.

1. Download the binary distribution from the downloads page
2. Install Tomcat application server (the DuraCloud service uses Tomcat 7)
    a. Please follow Tomcat configuration detailed on the Building From Source page.
3. Deploy the web applications
    a. Take the 3 .war files included in the binary distribution and copy them into the "webapps" directory under your tomcat installation.
    b. Start tomcat, as part of the startup process, tomcat will unpack the wars and deploy them
4. Initialize the DuraCloud applications
    a. Open the init.properties file found in the distribution package
    b. Edit the values that are contained in brackets "[...]" to be appropriate for your environment. This is the step where most problems tend to show up, so feel free to ask questions regarding what values need to be included in this file.
    c. A few notes on editing this file:
        - The value of [host] will almost certainly be: localhost
        - You will need to have your own Amazon S3 account in order to connect. The "username" and "password" in the init.properties file, in the case of Amazon, refer to the Access Key ID and the Private Access Key that are used to make API connections to Amazon Web Services.
        - You probably want to comment out (using '#') the section starting with 'durastore.storage-acct.1', unless you also want to create a Rackspace CloudFiles account.
        - The security user settings at the bottom allow you to indicate the username and password users accounts that can access your local DuraCloud, you'll use these credentials to log in to DuraCloud after initialization
    d. Execute the app-config jar, passing in the init.properties file as a parameter

```
java -jar app-config.jar init.properties
```

    e. The very last line of output from the execution of the app-config process should be "success". If that's not the case, look at the output more closely to determine what error may have occurred.
5. Log in to the application by going to http://localhost:8080/duradmin.
6. Congratulations, you are now running DuraCloud!

Building DuraCloud Software from Source

Introduction

> If you would prefer to install DuraCloud from a binary distribution, you can find instructions for that process here.

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 Windows Azure)
- DuraCloud REST API
- DuraBoss - this web application includes the reporter, executor, auditor, and manifest projects, each of which expose their own distinct set of calls via the DuraCloud REST API
- DurAdmin - this web application is the UI front end of DuraCloud, it provides users with a view into the information available from the other applications. DurAdmin uses the REST APIs of the other applications to communicate with them.
- 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 providers. As may be expected, these integrations are necessary for the system to be properly exercised. In order for DuraCloud to connect to these underlying providers, appropriate credentials must be provided as part of the application initialization step. It is recommended that you acquire the necessary storage provider credentials prior to attempting to set up DuraCloud. Only one storage provider is required to run DuraCloud.

The storage providers which are currently supported are listed here.

This guide lays out the steps necessary to begin using DuraCloud:

1. Build and deploy the DuraCloud web applications
2. Initialize the DuraCloud applications
3. Test your installation

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. Java: Version 7 required, Java 8 preferred
    a. The Oracle JDK is recommended for building DuraCloud, as this is the JDK used for DuraCloud testing.
2. Maven 3.x
3. Tomcat 7.x
4. Git

Setting up DuraCloud

> 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 and deploy the DuraCloud web applications

1. Check out latest stable release from Subversion repository

```
git clone --branch duracloud-3.5.0 https://github.com/duracloud/duracloud
```

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

    b. Add to $CATALINA_HOME/conf/server.xml

    Add the config attribute "URIEncoding" with value "UTF-8" to your Tomcat Connector. Your connector may look like the following:

```
<Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               URIEncoding="UTF-8" />
```

4. Start Tomcat

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

5. Configure Maven3
    a. Add tomcat user to $M2_HOME/conf/settings.xml. Make sure that the username and password used here match those included in the tomcat-users.xml file.

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

6. Build
    a. From top of the source tree, execute (note that this build runs unit tests, but not integration tests):

```
mvn clean install -DskipIntTests
```

7. Bonus, running Integration Tests
    a. Grab a copy of the file in the codebase under common-json/src/main/resources/test-config.json and place it somewhere on your system
    b. You'll need an account for each of the providers to be tested. Update the JSON file to include your credentials. These tests actually communicate with each storage provider and verify that the calls being made work properly
    c. Add an environment variable called DURACLOUD-TEST-CONFIG with the value being the full path to your updated credentials JSON file
    d. From the top of the source tree, execute

```
mvn clean install -pl integration
```

Initialize the DuraCloud applications

1. Use the application initialization (app-config) utiltiy to configure the deployed DuraCloud applications
    a. If you haven't already done a full build, you need to build the app-config utility. From the top of the source tree, execute (you can skip this if you made it through step 6 above):

```
mvn clean install -pl app-config
```

    b. Run the app-config utility from the //app-config/target directory

```
java -jar target/app-config-[VERSION]-driver.jar <init.props>
```

        i. The init.props file is a configuration file which specifies all of the information necessary for the DuraCloud applications to run. An example of this file can be found at //app-config/src/main/resources/init.props. This file will need to be updated to match your environment.
    c. When the app-config utility completes successfully, the last line of output printed to the console will be the word "success". If this is not the case, check that your configuration file includes the correct information.

Test your installation

1. Once all of the above steps have been completed, your DuraCloud should be ready to test.
    a. Go to http://localhost:8080/duradmin (change host or port as necessary).
    b. Log in using the credentials provided in your configuration file
    c. You should be able to view, add, update, and delete spaces and content in Spaces tab
2. Congratulations! You now have a functional DuraCloud installation.

Optional items

SyncTool installers

1. The installers for the Sync Tool (Windows, Mac, Linux) are built using BitRock InstallBuilder Enterprise, so this tool must be installed locally.
    a. Download and install BitRock InstallBuilder Enterprise version 8.5.2 or above
    b. Place the license file for InstallBuilder in the InstallBuilder installation directory
2. Update the maven settngs.xml file ($M2_HOME/conf/settings.xml) to include the path to the installbuilder executable
    a. Add a duracloud profile to include the InstallBuilder path, and include the duracloud profile in the set of active profiles

```
<profiles>
  <profile>
    <id>duracloud</id>
    <properties>

<installbuilder.executable.path>[FULL-PATH-TO-BITROCK-INSTALLBUILDER-EXECUT
ABLE]</installbuilder.executable.path>
    </properties>
  </profile>
</profiles>

...

<activeProfiles>
  <activeProfile>duracloud</activeProfile>
</activeProfiles>
```

    b. From the //synctoolui directory in the DuraCloud source tree, execute

```
mvn clean install -Pinstallers
```

    c. The installers can be found under the synctoolui/target directory

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

Building Java client packages

1. To create a distributable zip of the storeclient or reportclient which includes their dependencies, from within the project directory (//storeclient, //reportclient) run

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

2. The packaged zip will be found under the project's target directory

Auxiliary Tools

Purpose

The tools listed below are generally single-purpose instruments for affecting change to or collecting information about content stored in DuraCloud. There are a variety of reasons why a tool may be built to operate against the DuraCloud API. All of these tools are command-line only.

These tools are not part of the primary DuraCloud baseline. They are, instead, set aside as a set titled Auxiliary tools. These tools are built to serve a need, and are tested and used for the original purpose. However, these tools are not subject to the same production standards as the primary DuraCloud baseline, so they should be used carefully.

Tools

Prefix Update Tool

Download Prefix Update Tool

Description

The purpose of the Prefix Update Tool is to allow for content item prefix values to be changed. If a given file path captured in the content IDs of files transferred to DuraCloud is incorrect, this tool can be used to correct that path. If a set of files needs to be given a prefix after they have already been transferred to DuraCloud, this tool can be used for that purpose as well.

Command Line 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	
-s	--space-id	Yes	Yes	The ID of the DuraCloud space where content will be stored	
-u	--username	Yes	Yes	The username necessary to connect to DuraCloud	
-p	--password	Yes	Yes	The password necessary to perform writes to DuraStore	
-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
-o	--old-prefix	Yes	Yes	The prefix value that is to be replaced. Use "" to match all content items (if adding a prefix to all content items in a space is desired.)	
-n	--new-prefix	Yes	Yes	The prefix value that will be applied to all content items matching the old prefix. Use "" to simply remove the old prefix.	
-d	--dry-run	No	No	Indicates that no changes should be made to DuraCloud, but that the changes which would be made should be printed to the console. This allows you to see what the tool would do without changing anything.	

Example Usage

```
java -jar prefixupdatetool-1.0.0-driver.jar -h institution.duracloud.org -s
my-favorite-space -o C:/bobsfiles/Fall 2002 notes/ -n robert/notes/2002-fall/ -u
[username] -p [password]
```

The above command would cause the files stored in the space named "my-favorite-space" which have the existing prefix "C:/bobsfiles/Fall 2002/notes" to be updated such that the original prefix would be replaced with "robert/notes/2002-fall. For example a file in the space with the content ID: "C:/bobsfiles/Fall 2002 notes/board-meetings/2002-11-01-meeting-notes.txt" would be changed to "robert/notes/2002-fall/board-meetings/2002-11-01-meeting-notes.txt"".

Getting Started with DuraCloud Mill

This article describes the necessary steps for configuring and running the DuraCloud Mill. The DuraCloud Mill is a collection of applications that work in concert with DuraCloud to perform a variety of bulk processing tasks including audit log writing, manifest maintenance, duplication and bit integrity checks. While the Mill has been designed to run in an auto-scalable environment such as AWS EC2, it can also be run as a suite of stand-alone applications on any machine with sufficient computing resources. This article only describes the various applications and how to configure them; it does not cover approaches to leveraging AWS autoscaling and supporting dev opts tools such as Puppet.

If you are not yet familiar with the DuraCloud Mill please refer to the DuraCloud Architecture document, which describes the purpose and primary components of the Mill.

Download, Build, Configure

1. To start, clone and build the latest release of the mill

```
git clone https://github.com/duracloud/mill.git
cd mill
mvn clean install
```

2. Create the empty mill database, add credentials and than create the schema using mill/resources/mill.schema.sql (found in the mill code baseline.)
3. Create a configuration file.
    a. Now that you've built the mill and created the database, we need to set up a configuration file that can be used my the various components of the system. A template of this configuration file can be found in the base line at mill/resources/mill-config-sample.properties
        i. Copy and rename the file to mill-config.properties
        ii. Configure the database connections to the mill database as well as the management console database:

```
###################
# MILL DATABASE
###################
# Config for mill database.
mill.db.host=[fill in]
mill.db.port=[fill in]
mill.db.name=[fill in]
# User must have read/write permission
mill.db.user=[fill in]
mill.db.pass=[fill in]
#Turn this feature on to generate the milldb
#hibernate.hbm2ddl.auto=update
###################
# MC DATABASE
###################
# Config for the management console database - used to retrieve
accounts and storage provider credentials
db.host=[fill in]
db.port=[fill in]
db.name=[fill in]
# User must have read permission
db.user=[fill in]
db.pass=[fill in]
```

Configuring and Running Workman

Workman is responsible for reading tasks off a set of queues, delegating them to task processors, and removing them once they have reached a completed state. In the case of failures, tasks are retried three times before they are sent to the dead letter queue. A single instance of Workman can run multiple tasks in parallel. How many tasks depends on the **max-workers** setting in the mill-config.properties file. It is also safe to run multiple instances of workman a single machine as well as multiple. We recommend running a single instance of workman on each machine instance, setting the max-workers setting in accordance with the available resources.

1. Queue Names refer to the AWS SQS queue names defined in your account. You must create and configure the following queues as defined in the queue section by replacing the brackets ([]) with names.

```
#########
# Queues
#########
queue.name.audit=[]
queue.name.bit-integrity=[]
queue.name.dup-high-priority=[]
queue.name.dup-low-priority=[]
queue.name.bit-error=[]
queue.name.bit-report=[]
queue.name.dead-letter=[]
```

2. For a given instance of workman you must specify which queues will be consumed and the order in which they will be read. In other words, a given instance of workman can focus on specific kinds of tasks. It can also decide which tasks have a higher priority. In this way, instances of workman can be configured to work on hardware configurations that are suitable to the load and kinds of tasks they will need to bear. Make sure you use the above defined keys rather than the queue names themselves.

```
## A comma-separated prioritized list of task queue keys (ie do not use the
## concrete aws queue names - use  queue.name.* keys) where the first is highest
priority.
## The first items in the list have highest priority; the last the lowest.
queue.task.ordered=[]
```

3. As we mentioned before, max-workers sets the number of task processing threads that can run simultaneously.

```
# The max number of worker threads that can run at a time. The default value is
5.
max-workers=[]
```

4. The duplication policy manager writes policies to an S3 bucket. Both the loopingduptaskproducer and workman use those policies for making decisions about duplication.

```
# The last portion of the name of the S3 bucket where duplication policies can be
found.
duplication-policy.bucket-suffix=duplication-policy-repo
# The frequency in milliseconds between refreshes of duplication policies.
duplication-policy.refresh-frequency=[]
```

5. You can also set the workdir which defines where temp data will be written as well as notification.recipients.

```
# Directory that will be used to temporarily store files as they are being
processed.
workdir=[]
# A comma-separated list of email addresses
notification.recipients=[]
```

6. Once these settings are in place you can run workman by simply invoking the following java command:

```
java -Dlog.level=INFO -jar workman-{mill version here}.jar -c
/path/to/mill-config.properties
```

Configuring and Running Duplication

Once you have an instance of workman running you can perform an explicit duplication run. The spaces that have been configured with duplication policies (see the Mill Overview for details) will generate duplication events when the audit tasks associated with them are processed. If you add a new duplication policy to a new space that already has content items, you'll need to perform a duplication run to ensure that those new items get duplicated. The loopingduptaskproducer fulfills this function. Based on the set of duplication policies, it will generate duplication tasks for all matching spaces. It will keep track of which accounts, spaces and items have been processed in a given run so it does not need to run in daemon mode. It will run until it has reached the max number of allowable items on the queue and then it will exit. The next time it is run, it will pick up where it left off. You may want to dial down the max queue size in the event that you have so many items and so little computing power to process them with that you may exceed the maximum life of an SQS message (which happens to be 14 days). It should also be noted here that items are added roughly one thousand at a time for each space in a round-robin fashion to ensure that all spaces are processed in a timely way. This strategy ensures that small spaces that are flanked by large spaces are processed quickly. **It is also important that only one instance of loopingduptaskproducer is running at any moment in time.** Two settings to be concerned with when it comes to the looping dup task producer:

```
############################
# LOOPING DUP TASK PRODUCER
############################
# The frequency for a complete run through all store policies. Specify in hours (e.g.
3h), days (e.g. 3d), or months (e.g. 3m). Default is 1m - i.e. one month
looping.dup.frequency=[]
# Indicates how large the task queue should be allowed to grow before the Looping Task
Producer exits.
looping.dup.max-task-queue-size=[]
```

The program can be run using hte following command:

```
java -Dlog.level=INFO -jar loopingduptaskproducer-{mill version here}.jar -c
/path/to/mill-config.properties
```

Configuring and Running Bit Integrity

Bit integrity runs works similarly to the duplication runs. **loopingbittaskproducer** has similar settings as those mentioned above as well as two others, looping.bit.inclusion-list-file and looping.bit.exclusion-list-file. These two config files let you be more surgical in what you decide to include and exclude from your bit integrity run and function similarly to the duplication policies. The important thing to note here is that if there are not entries in the inclusion list all accounts, stores, and spaces are included. **It is also important that only one instance of loopingbittaskproducer is running at any moment in time.**

```
############################
# LOOPING BIT TASK PRODUCER
############################
# The frequency for a complete run through all store policies. Specify in hours (e.g.
3h), days (e.g. 3d), or months (e.g. 3m). Default is 1m - i.e. one month
looping.bit.frequency=[]
# Indicates how large the task queue should be allowed to grow before the Looping Task
Producer quits.
looping.bit.max-task-queue-size=[]
# A file containing inclusions. Expressions will be matched against the following
path: /{account}/{storeId}/{spaceId} and should have the same format. You can use an
asterix (*) to indicate all.
# For example,  to indicate all spaces name "space-a"  in the "test" account across
all providers you would add a line like this:
#    /test/*/space-a
# You may also comment lines by using a hash (#) at the beginning of the line.
looping.bit.inclusion-list-file=[]
# A file containing exclusions as regular expressions using the same format as
specified for inclusions.
looping.bit.exclusion-list-file=[]
```

The program can be run using the following command:

```
java -Dlog.level=INFO -jar loopingbittaskproducer-{mill version here}.jar -c
/path/to/mill-config.properties
```

Configuring the Audit Log Generator

Audit logs are initially written to the Mill database, then transitioned to a single AWS bucket for safekeeping. DuraStore (on the DuraCloud side) needs to know about this bucket in order to present these logs to API and DurAdmin users. The auditloggenerator application is responsible for reading from the database and writing these log files. The audit_log_item table in the database holds the audit events until the auditloggenerator writes and securely stores theses files in S3. Once written, the auditloggenerator will flag audit events as written as well as delete anything over thirty days old. Once the audit log generator has written and purged everything there is to be written and purged, it will exit. It is recommended to run this process on a cron job to ensure that audit logs are processed in a timely way. **It is also important that only one instance of auditloggenerator is running at any moment in time.**

```
# The global repository for duracloud audit logs
audit-log-generator.audit-log-space-id=[]
```

The program can be run using the following command:

```
java -Dlog.level=INFO -jar auditloggenerator-{mill version here}.jar -c
/path/to/mill-config.properties
```

Configuring the Manifest Cleaner

Finally the manifest-cleaner program is responsible for clearing out deleted manifest items from the database. When items are deleted from the manifest, initially they are flagged as deleted rather than being deleted from the database. We do it this way to ensure that we can correctly handle add and delete messages that are delivered out of order (since SQS does not guarantee FIFO). However, eventually we do want to purge deleted records to keep our database from growing too large. So the manifest cleaner simply checks for "expired" deleted items and flushes them out of the db. It has just one setting (in a addition to the database configuration information we mentioned above). We don't recommend specifying any time less than 5 minutes.

```
###################
# MANIFEST CLEANER
###################
# Time in seconds, minutes, hours, or days after which items deleted items should be
purged.
# Expected format: [number: 0-n][timeunit:s,m,h,d]. For example 2 hours would be
represented as 2h
manifest.expiration-time=[]
```

The program can be run using hte following command:

```
java -Dlog.level=INFO -jar manifest-cleaner-{mill version here}.jar -c
/path/to/mill-config.properties
```

Related articles