# Migration API – Software Design Document

Currently, the user’s data resides in Cleversafe and/or Cloudian archives. In order to facilitate the transfer of 1) Existing high value datasets from Cleversafe, and 2) Stale, unneeded data to Glacier, via AWS S3, a new set of Migration API shall be introduced. This API shall be used to migrate users’ data from its current archive to any AWS or third party S3 bucket. It will cover the following use cases; transfer from Cleversafe to Cloudian, transfer from Cleversafe to AWS S3 Glacier, and transfer from Cloudian to AWS S3 Glacier

# API Design

At a high-level, the following tasks shall be performed.

* Leverage the existing AWS S3 download API to download the data using AWS S3 API
* Introduce a scheduler job to repoint the archive location metadata.
* Attach object metadata (uuid and user\_id) to the new object in AWS S3.
* Delete from the source once file verification is performed.

The following shall be taken into consideration.

* The destination S3 configuration and account credentials shall be configured upfront.
* Run migration-scheduler on the backup server to perform the download.
* Incorporate auto rotation of keys for AWS S3 buckets using AWS Secrets manager. Each bucket will have an access key/secret key pair, and will be auto rotated via cloud watch and lambda function. Secrets manger access key will be stored and updated in the SYSTEM ACCOUNT table via a DME scheduler job. This key will be used to obtain the individual bucket access keys through the Secrets manager API.
* For the actual data transfer, make the code configurable so that we can plug-in the StrongLink API or some other third-party vendor tools to perform the data transfer in the future.

## Single File Migration API

* Data Object Migration Transfer from the current archive to the destination S3 archive

## Collection Migration API

* Collection Migration Transfer from the current archive to the destination S3 archive

# Detailed Design

This section provides the detailed design of the changes to code base to support the new functionality to migrate an existing archive to any third party S3 provider.

## Rest API

* **hpc-dto/HpcDataMigration.xsd**
  + The model (DTO) for the Data Migration API will be added to specify the download destination. (Specify S3 config Id instead of supplying credentials.) Add HpcMigrationRequestDTO to specify the S3 config id. Add HpcMigrationResponseDTO to return the taskId.
* **hpc-ws-rs-api/HpcDataMigrationRestService.java**
  + Add migrateDataObject() method based on the API design above, and bind it to /dataObject/{path:.\*}/migrate endpoint. This method implements the single file transfer API from the current archive to the destination S3 archive.
  + Add migrateCollection() method based on the API design above, and bind it to /collection/{path:.\*}/migrate endpoint. This method implements the collection transfer from the current archive to the destination S3 archive.
* **hpc-ws-rs-impl/HpcDataMigrationServiceImpl.java**
  + Implement the 2 new API methods, using the Data Migration model.

## Business / Application Services

* **hpc-bus-service-api/HpcDataMigrationBusService.java**
  + Add migrateDataObject(), migrateCollection() methods.
* **hpc-bus-service-impl/HpcDataMigrationBusServiceImpl.java**
  + Implement the migrateDataObject() method. Obtain the destination bucket account credentials, call app service layer to submit a migration download task and return the task ID.
  + Implement the migrateCollection() method. Call the migration app service API to obtain the destination bucket account credentials, submit a collection migration download task and set and return the task ID.
* **hpc-app-service/HpcDataMigrationService.java**
  + Add migrateDataObject(), migrateCollection() methods.
* **hpc-app-service-impl/HpcDataMigrationServiceImpl.java**
  + Add method to obtain bucket credentials from S3 archive config id.
  + Implement method migrateDataObject() and migrateCollection() methods which create an async download task in new migration table.
  + The actual transfer (download) for migration tasks shall be picked up from the new scheduler job.

## Integration

* **HpcDataTransferProxy**
  + Add API to access the AWS Secret manager to obtain the latest bucket credentials

## DAO

* **hpc-dao-impl/HpcDataMigrationDAOImpl.java**
  + Add methods to upsert record to migration data object download task and the corresponding result table.

## Scheduler

* **hpc-scheduler/HpcScheduledTasksImpl.java**
  + New scheduler job needs to be created to pick up the migration download tasks and initiate the download.
  + Add completeMigrationTasks() to periodically check for completion of data object download migration tasks and update iRODS **s3 archive configuration id**, add **migration\_date(s)** metadata. The metadata, **data\_transfer\_status** shall toggle to DEEP\_ARCHIVE if the destination is AWS S3, assuming that we will always be transitioning to GLACIER. If the destination archive is Cloudian, this will remain as ARCHIVED. Verify the downloaded data object and delete from the source archive.
  + Add new scheduler job to auto rotate the Secret manager access key stored in SYSTEM ACCOUNT table.

# Questions/Open items

* What about collections that has mixed provider some data object in Cleversafe, some already in Cloudian?
  + We’ll need to check if the current S3 archive configuration id is the same as the destination S3 archive config id, ignore.
* Need to distinguish between migration downloads vs. regular downloads for processing so it gets picked up from different servers.
  + Keep the migration download table separate from existing tables.
* Download task result table growing is a concern.
  + Reports might be using these tables, so we need to design archival of these tables carefully.
  + Keeping the migration download table separate from existing table.
* How do we validate the downloaded file?
* Shall we add new event for data migration completed?
  + No need to add event type for now, we shall use the DATA\_TRANSFER\_DOWNLOAD\_COMPLETED event.
* Using Registration API instead of Download API.
  + Original system metadata will be lost such as data transfer started.
  + Will lose original source such as Globus etc.
  + Work is required to supply exising user metadata.