*HPC Data MANAGEMENT*

User guide

Version *1.8*

*01/23/2019*

TABLE OF CONTENTS

[1 Purpose 6](#_Toc532852243)

[2 Introduction 6](#_Toc532852244)

[2.1 What is HPC DME? 6](#_Toc532852245)

[2.2 Intended users 6](#_Toc532852246)

[2.3 Accessing HPC DME 6](#_Toc532852247)

[3 Pre-Requisites to access these HPC DME APIs 7](#_Toc532852248)

[3.1 Account with HPC DME 7](#_Toc532852249)

[3.2 Using Globus for asynchronous data transfers 7](#_Toc532852250)

[3.3 Pre-requisites for HPC DME utilities and CLI 9](#_Toc532852251)

[3.3.1 Java 9](#_Toc532852252)

[3.3.2 Get the HPC DME CLI 9](#_Toc532852253)

[4 HPC DME Overwiew 10](#_Toc532852254)

[4.1 Business rules and characteristics of collections 11](#_Toc532852255)

[4.2 Business rules and characteristics of data objects 12](#_Toc532852256)

[4.3 Metadata 12](#_Toc532852257)

[4.4 User authentication 14](#_Toc532852258)

[5 HPC DME Core FUNCTIONALITIES 14](#_Toc532852259)

[5.1 Group administrator functions 15](#_Toc532852260)

[5.1.1 Enroll user into HPC DME (create) 15](#_Toc532852261)

[5.1.2 Add or remove user(s) from a group 15](#_Toc532852262)

[5.1.3 Delete data object by group administrators only 15](#_Toc532852263)

[5.2 Register/Upload functions 15](#_Toc532852264)

[5.2.1 Register collection with its metadata into HPC DME(create and update) 15](#_Toc532852265)

[5.2.2 Register data object along with metadata into HPC DME (create and update) 15](#_Toc532852266)

[5.3 Assign permissions on HPC DME collections or data objects 15](#_Toc532852267)

[5.4 Search functions 16](#_Toc532852268)

[5.4.1 Find collection by metadata 16](#_Toc532852269)

[5.4.2 Find data object by metadata 16](#_Toc532852270)

[5.5 Get collection by path 17](#_Toc532852271)

[5.6 Get data object by path 17](#_Toc532852272)

[5.7 Download functions 17](#_Toc532852273)

[5.7.1 Download data object/file to another globus endpoint 17](#_Toc532852274)

[5.7.2 Download data object/file to local file system 17](#_Toc532852275)

[6 Executing HPC DME API with Command Line Client 18](#_Toc532852276)

[6.1 Seting the HPC DME CLI 18](#_Toc532852277)

[6.2 Running HPC DME Command line Client 19](#_Toc532852278)

[6.2.1 Interactive Mode: 19](#_Toc532852279)

[6.2.2 Batch Mode 20](#_Toc532852280)

[6.3 Error handling 33](#_Toc532852281)

[7 Executing HPC DME API with Command line utilities 33](#_Toc532852282)

[7.1 One time setup 33](#_Toc532852283)

[7.2 Generate a Data management API token 33](#_Toc532852284)

[7.3 Generate a Globus token 33](#_Toc532852285)

[7.4 Register a collection 34](#_Toc532852286)

[7.5 Register a data object 34](#_Toc532852287)

[7.5.1 Synchronously (From file system) 34](#_Toc532852288)

[7.5.2 Asynchronously (From Globus) 34](#_Toc532852289)

[7.6 Register a Directory 35](#_Toc532852290)

[7.6.1 Synchronously (from file system) 35](#_Toc532852291)

[7.6.2 Asynchronously (from Globus) 35](#_Toc532852292)

[7.7 Get a collection metadata 36](#_Toc532852293)

[7.8 Get a dataobject metadata 36](#_Toc532852294)

[7.9 Download a dataobject synchronously 36](#_Toc532852295)

[7.10 Download a Collection or dataobject Asynchronously 36](#_Toc532852296)

[7.11 Download a Collection or DataObject to an s3 bUCKET 36](#_Toc532852297)

[7.12 Delete a Dataobject From the Archive 37](#_Toc532852298)

[7.13 Delete a Collection from the Archive 37](#_Toc532852299)

[7.14 Move/Rename a file or collection in the Archive 38](#_Toc532852300)

[8 Executing HPC DME API with CURL 38](#_Toc532852301)

[8.1 Setting up the environment 39](#_Toc532852302)

[8.2 Executing CURL commands 39](#_Toc532852303)

[8.2.1 Registering a new user 39](#_Toc532852304)

[8.2.2 Registering a collection 39](#_Toc532852305)

[8.2.3 Registering a data object 39](#_Toc532852306)

[8.2.4 Search for collection 40](#_Toc532852307)

[8.2.5 Search for data Object 40](#_Toc532852308)

[8.2.6 Assign Permissions 40](#_Toc532852309)

[8.2.7 Download data object to a Globus endpoint 40](#_Toc532852310)

[8.2.8 Download data object to a file system 41](#_Toc532852311)

[8.2.9 Download data object to an S3 bucket 41](#_Toc532852312)

[8.2.10 Download collection to an S3 bucket 41](#_Toc532852313)

[8.2.11 Delete Data Object (by group administrators only) 42](#_Toc532852314)

[8.2.12 Move a collection 42](#_Toc532852315)

[8.2.13 Move a data object 42](#_Toc532852316)

[8.2.14 Bulk move 42](#_Toc532852317)

[9 Executing HPC DME API with Web Client 43](#_Toc532852318)

[9.1 Dashboard and login/logout 43](#_Toc532852319)

[9.1.1 Log In 43](#_Toc532852320)

[9.1.2 Log Out 43](#_Toc532852321)

[9.1.3 Dashboard 44](#_Toc532852322)

[9.2 Search functions 44](#_Toc532852323)

[9.2.1 Google Style Search 45](#_Toc532852324)

[9.2.2 Basic Search 45](#_Toc532852325)

[9.2.3 Advanced Search 45](#_Toc532852326)

[9.2.4 Saved Searches 45](#_Toc532852327)

[9.2.5 Bookmark Function 46](#_Toc532852328)

[9.3 BROWSE functions 48](#_Toc532852329)

[9.4 Register a collection 50](#_Toc532852330)

[9.5 Register a Data File 53](#_Toc532852331)

[9.6 Bulk Register (Globus) 55](#_Toc532852332)

[9.7 SHARING functions 56](#_Toc532852333)

[9.8 Update metadata in non-batch mode 57](#_Toc532852334)

[9.9 Download collection or file from archive 58](#_Toc532852335)

[9.9.1 Share a Globus End Point with HPCDME Globus Group 58](#_Toc532852336)

[9.9.2 Download a Collection of Data Files 60](#_Toc532852337)

[9.9.3 Download a Single Data File 61](#_Toc532852338)

[9.10 Notifications, task statuses and reports 63](#_Toc532852339)

[9.10.1 Notifications Management 63](#_Toc532852340)

[9.10.2 Task Statuses 63](#_Toc532852341)

[9.10.3 Reports Management 67](#_Toc532852342)

[9.11 Group administrator functions 68](#_Toc532852343)

[9.11.1 User 69](#_Toc532852344)

[9.11.2 Group 70](#_Toc532852345)

[9.11.3 Delete a Data Object and Associated Metadata 72](#_Toc532852346)

[10 APPENDIX A – sample collection input 74](#_Toc532852347)

[11 APPENDIX B – sample data file input 74](#_Toc532852348)

[12 Appendix c – sample permissions input 74](#_Toc532852349)

[13 APPENDIX D – Executing HPC DME API with Soap UI 74](#_Toc532852350)

[13.1 Prerequisites 74](#_Toc532852351)

[13.2 Access HPC DME with Soap UI 76](#_Toc532852352)

[13.3 Use SOAP UI to perform common use scenarios 76](#_Toc532852353)

[13.3.1 Register a collection (Project or dataset) 76](#_Toc532852354)

[13.3.2 Register a single data file/object into storage archive asynchronously 77](#_Toc532852355)

[13.3.3 Register a single data file/object into storage archive synchronously 80](#_Toc532852356)

[13.3.4 Perform update on a metadata attribute 81](#_Toc532852357)

[13.3.5 Subscribe to a known event 82](#_Toc532852358)

[13.3.6 Generate a report 83](#_Toc532852359)

[13.3.7 Update/assign permission 84](#_Toc532852360)

[13.3.8 Perform simple search functions 85](#_Toc532852361)

[13.3.9 Download a data file/object to Globus share 87](#_Toc532852362)

[13.3.10 Download a data file/object to a local directory 88](#_Toc532852363)

[14 APPENDIX E HPC DME CLI properties 88](#_Toc532852364)

# Purpose

The user guide describes the core High Performance Computing (HPC) Data Management Environment (DME) functional capabilities, i.e., registering (uploading), transferring, sharing, searching, downloading collections (i.e., datasets, projects, or folders) and data objects (i.e. files). It further outlines various means of performing such actions from an end user’s perspective: i.e., executing core APIs from Web, command line client and utilities, and CURL. The user guide also includes details of the prerequisites needed for setting up the HPC DME clients**.**

# Introduction

## What is HPC DME?

The HPC DME, High Performance Computing Data Management Environment, is a highly adaptable and an open ended data storage environment supporting storage and management of data, produced from high performance computing systems. HPC DME provides capabilities for storing, managing, transferring and sharing these data across different storage systems securely and efficiently.

Users can store data objects (also referred to as Data files) for a long term on HPC DME, share and transfer their data such that they do not have to redistribute or maintain copies of the data on other systems by eliminating the data integrity issues. HPC DME stores and associates user defined metadata to any registered data at different levels of data life cycle, enabling the environment to not only ~~to~~ help identify the data but also enhance the search capabilities and to be able to attach a value factor to each dataset.

## Intended users

The HPC DME has been built to cater to the data storage and data management needs of NCI cancer community. Any user with a valid HPC DME user account can run its client interface. *Note: HPC DME supports only NIH Active Directory account holders at this point of time*.

## Accessing HPC DME

The HPC DME APIs, associated servers and infrastructures are deployed on Development, UAT and Production tiers for ongoing development, user acceptance and exploration usage with limited storage allocated and CleverSafe vault customized through the Frederick ITOG team. Data storage needs will be assessed, estimated and monitored as and when a new user or group of users want to use the HPC DME pilot/s.

HPC DME can be accessed through its Service APIs. These HPC DME Service APIs are developed based on RESTful implementation of standard. Therefore, these APIs can be accessed by any interface that is compatible with REST standard. For example, these Service APIs can be accessed through Java, Perl, Ruby, Curl, JavaScript languages programmatically or through UI clients like SOAP UI, REST Client (See [Appendix D: Using Soap UI to execute HPC DME APIs](#_APPENDIX_D_-)).

We have also provided an example implementation to access these HPC DME APIs using a web GUI, and command line utilities for shell users. Detailed information on accessing these service APIs will be the focus of this User Guide.

The following is the HPC DME Service API URL for the production tier:

<https://hpcdmeapi.nci.nih.gov:8080>

The following is the HPC DME Service API URL for the User Acceptance Test (UAT) tier:

<https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server>

# Pre-Requisites to access these HPC DME APIs

## Account with HPC DME

You will first need to register with HPC DME for an account. Please consult HPC DME administrator for an account at [HPC\_DME\_Admin@nih.gov](mailto:HPC_DME_Admin@nih.gov) after obtaining your group or DOC approval for using the HPC DME. If you are a HPC DME administrator, please see HPC Server API specifications for user registration details.

## ­Using Globus for asynchronous data transfers

HPC DME supports transferring data from a Globus endpoint (<https://docs.globus.org/how-to/>) to HPC DME Archive storage, and from HPC DME Archive storage to a Globus endpoint. Though using Globus is not necessarily a prerequisite to transfer data files (objects) into and from the Cleversafe object store or any other storage medium like Isilon storage system, it is recommended for transferring large data objects (greater than or equal to 5 GB) asynchronously.

The HPC DME environment uses a Globus Group pooling mechanism to perform asynchronous transfer from and to Cleversafe. If you would like to register data that resides on a Globus endpoint into the archive, you will first grant READ permission to the Globus Group your DOC is associated with before your registration. Similarly, if you would like to download data from the archive to a Globus endpoint, you will grant the Globus Group WRITE permission to the shared Globus Endpoint.

Note that sharing a Globus endpoint is only available if you have Globus Plus account. Biowulf, Frederick, and CBIIT endpoints have Globus Plus accounts, while the Globus connect personal endpoint does not have this privilege by default. For more information about sharing a Globus endpoint and granting access permission, please review [How To Share Data Using Globus](https://docs.globus.org/how-to/share-files/). You can also contact [staff@hpc.nih.gov](mailto:staff@hpc.nih.gov) if you would like to have a Globus Plus account on your server or laptop. Globus does not support sharing individual files. If you have a single file to register, please create a folder for it and share the folder.



Here are the Globus Groups for different HPC DME tiers:

|  |  |  |
| --- | --- | --- |
| Globus Group | Applicable DOC | Applicable HPC DME Tier (s) |
| HPCDME-PROD-App-Accts-Pool-FNLCR | Sequencing Facility @FNLCR | Production |
| HPCDME-PROD-App-Accts-Pool-HITIF | HiTiF, CCR | Production |
| HPCDME-PROD-App-Accts-Pool-CCBR | CCBR | Production |
| HPCDME-PROD-App-Accts-Pool-CCR\_SBL | SBL, CCR | Production |
| HPCDME-PROD-App-Accts-Pool-Default | All other DOCs | Production |
| HPCDME-DEV-App-Accts-Pool-FNLCR | Sequencing Facility @FNLCR | UAT |
| HPCDME-DEV-App-Accts-Pool-HITIF | HiTiF, CCR | UAT |
| HPCDME-DEV-App-Accts-Pool-CCBR | SBL, CCR | UAT |
| HPCDME-DEV-App-Accts-Pool-CCR\_SBL | SBL, CCR | UAT |
| HPCDME-DEV-App-Accts-Pool-Default | All other DOCs | UAT |

## Pre-requisites for HPC DME utilities and CLI

### Java

Download and install Java 1.8.x

<https://www.java.com/en/download/>

Please visit the following link for installation instructions

<https://www.java.com/en/download/help/index_installing.xml>

After successful installation of Java, update PATH with Java executable folder (Ex: c:/jdk1.8\_13/bin)

<https://java.com/en/download/help/path.xml>

Open cmd/shell window and type “java –version” to make sure it displays the version information you installed.

### Get the HPC DME CLI

Clone HPC DME Utils into your local machine.

Git clone <https://github.com/CBIIT/HPC_DME_APIs>

Navigate to HPC\_DME\_APIs/utils folder. Open README.md and follow the instructions.

Note: For windows users, you have to first install [Cygwin](https://www.cygwin.com/) to use the command line utilities. You can also open a terminal cmd to run the java HPC DME CLI.

# HPC DME Overwiew

The HPC data management environment provides a high-reliability storage model for underlying collections including a collection registration system, and an API for transfer of data objects. The data object registration associates a label with a given managed data file or folder and captures extensible metadata for the managed data object.

The HPC data management environment provides a number of application programming interfaces (APIs) to operate and interact with it. At a high level, there are two important components in HPC DME:

1. Metadata management: HPC DME by default integrates with iRODS iCAT instance to manage metadata and its security for both collections and data objects.

2) Data transfer: HPC DME can transfer data directly from a file system (synchronous registration), or use Globus to perform asynchronous data transfer between Globus endpoint and CleverSafe storage (asynchronous registration). Synchronous registration integrates with S3 API to transfer data from local file system to HPC DME archive. The current approach is to stream data files (objects) from local/designated file directory to the application server, and then stream that to the Cleversafe archive.

HPC DME’s pluggable architecture allows both these implementations to be replaced with alternatives easily while keeping its APIs unchanged. The basic feature of HPC DME is to help users in registering and uploading their data to the HPC DME archive storage and managing it. HPC DME archive storage can be a permanent storage for the users’ data and be used as a platform to search, manage and transfer the data to other storage systems and also to share with other collaborators or users. Each data object is stored along with its required and user defined metadata associated with it. The associated metadata can be used as search criteria to identify dataset(s).

The HPC data management implementation provides users flexibility to define collections and data objects. Collections in HPC DME can be Projects, Datasets, etc. and data objects are the data files. Data object is defined as an individual file. Collections can be of different types for example, a dataset can be linked to a project from any domain such as sequencing project, imaging study or any data analysis project etc., therefore a project can be an entity (umbrella project) indicating a scientific ‘study’, or simply a folder. To summarize, HPC DME supports metadata registration to connect unstructured data files or objects and give it a minimal structure required to precisely identify a dataset. Metadata may be attached to files and collections (equivalent of sub-directories). Though each laboratory or NCI business unit may adopt or engage with a variety of biomedical/clinical research and produce many forms of data, HPC DME does enable flexibility for each DOC to have their own way of organizing/structuring how their data/metadata is presented or preserved.

The HPC DME database will store metadata in the form of “triples”. Each metadata element/entity/entry will be a triple consisting of an attribute field, a value field, and a unit field. The content of each of these fields can be independently defined and applied. Metadata may be user-defined or applied programmatically.

## 4.1 Business rules and characteristics of collections

* Each collection can be referred to with a logical path (“collection\_path”) which is unique across HPC DME.
* Irrespective of the logical path, each collection is associated with UUID metadata attribute to uniquely identify itself.
* A collection can be any category of business domain entities such as: Project, Dataset, Folder, Samples, Runs etc. More types can be configured in the system easily through metadata configuration. Please consult your HPC DME system administrator.
* Users can create/register multiple collections with HPC DME.
* Each collection can have its own required set of metadata which needs to be submitted at the time of registration. This required metadata can be configured by a system administrator with input from each DOC’s group administrator.
* Users have an option of adding new metadata variables to the required metadata associated with each collection.
* A collection can contain one or more data objects, or child collections. This is represented using the logical path of a given collection and may be achieved with user desired validation rules.

## 4.2 Business rules and characteristics of data objects

* A data object can be a single data file or a compressed file.
* Each data object can be referred with a logical path (“object\_path”) which is unique across HPC DME.
* A parent collection must be registered first with HPC DME before registering a data object.
* Irrespective of the logical path, each data object is associated with UUID metadata attribute to uniquely identify itself.
* Each data object can have associated required metadata which must be submitted at the time of data object registration.
* Users have an option of adding new metadata variables to the required metadata set, associated with each data object.

To emphasize, HPC DME supports DOC-dependent organization of their biomedical data/metadata, user defined collection types, more importantly allows and fully supports specific validation rules (policies) when they are registered into the archive. It is up to each DOC to come up, agree upon and determine how these policies will be enforced, standardized, to certain extent. Some DOC groups may script or automate the generations of their actual data/metadata files in their LIMS system to enable seamless integration between LIMS and HPC DME with little human intervention, verifying against all rules defined in the policy file.

## 4.3 Metadata

Metadata is defined as the data about the data. It is the information describing the actual data such as the date and origin of creation, its contents, its condition, its processing, or its associations to other objects. Metadata is employed to make data searches faster, more specific and also enable and promote data sharing among scientists.

HPC DME collects metadata for each collection and data object, registered and stored in a database along with the associations. HPC DME collects two kinds of metadata related to a collection or a data object, namely, administrative and center/division specific. The administrative metadata is the required set of information which needs to be submitted at the time of registration with HPC DME. New metadata variables can be added to both administrative and center/division specific metadata sets after a user obtains proper authorizations and permissions. The metadata can also be updated by authorized users through API or command line client.

HPC DME supports user defined collection types. Some of the collections can be Project, Sample, Study, Dataset. Collections types should be defined in the system policy file to enforce validation rules. To create new collection types, please submit the following information to HPC DME System Administrator.

Collection type name and its metadata are in the following format:

{

"attribute":"<Name of the attribute>",

"mandatory":<true/false>,

"defaultValue":"<Value>",

"collectionTypes":<Collection Type name>,

"validValues":[

"<Value1>",

"<Value2>"

],

"ruleEnabled":true,

"DOC":"<DOC-NAME>"

}

## 4.4 User authentication

After registering with HPC DME, the users can login into HPC DME using their user id. The HPC DME validates the user passwords through the NIH AD account.

# HPC DME Core FUNCTIONALITIES

HPC DME Service API is a RESTful interface. This interface is communicated over HTTPS with the same HTTP verbs (GET, POST, PUT, DELETE) that web browsers use to retrieve web pages and to send data to remote servers. HPC DME Service API can be categorized into different sections. Please see HPC DME Server API specifications for details on each of these functions.

HPC DME APIs are developed on RESTful implementation of standard, so any client interface can be used to access these APIs confirming REST standards. In this user guide, we provide a web interface and command line utilities to access the HPC DME APIs. The web interface is intended for users who are not familiar with command line. The command line utilities are intended for users who are familiar with the shell and consists of: a) The HPC DME CLI which is a java package for bulk processing and permission assignment on Linux and windows users. b) The command line utilities are used as wrapper for most common register and download operations. c) The curl utilities, described in other section in this document, show how to call the HPC DME API if the user needs finer control over calling the APIs.

In a typical scenario, users (producers and/or consumers) with the need of searching, browsing, sharing, and downloading data, will use the Web GUI. Group administrators, as the data owner for a DOC, will also use the Web GUI to manage their groups and enroll users. The command line utilities will be mostly used by data producers to archive bulk amount of data.

All these components will be detailed further below to explain how end users may use various clients, specifically the Web GUI, to perform needed and supported operation for managing archived data/metadata of their interest. If you are planning to use Globus for data transfer, please refer to section [3.2 on how to setup your Globus account.](#_Globus_Account_for)

## Group administrator functions

### [Enroll user into HPC DME](#_Enroll_User) (create)

[Adding a User via the GUI](https://wiki.nci.nih.gov/x/-YapFg)

[Enroll a User via CURL](#_Registering_a_new)

### Add or remove user(s) from a group

[Updating a Group via the GUI](https://wiki.nci.nih.gov/x/BYepFg)

### Delete data object by group administrators only

*This delete data object and associated metadata function is implemented to support rare case scenarios where an end user may make mistakes in registering the wrong files. After consulting the users, only a group administrator with a data ownership to the object file will be allowed to perform this function to enforce and maintain data integrity. Moreover, associated metadata will be deleted from the database.*

[Deleting a File via the GUI](https://wiki.nci.nih.gov/x/7oapFg)

## Register/Upload functions

### [Register collection with its metadata into HPC DME](#_Register_Project)(create and update)

[Registering Data via the GUI](https://wiki.nci.nih.gov/x/aQmKFg)

[Register Collection via Command Line Utility](#_7.4_Register_a)

[Register a Directory via Command Line Utility](#_Register_a_Directory)

[Register Collection via CURL](#_Registering_a_collection)

### [Register data object along with metadata into HPC DME](#_Register_Dataset) (create and update)

In case of asynchronous dataObject registration using Globus, you have to provide the endpoint Unique Universal ID (UUID) to the API. You can retrieve a Globus endpoint UUID by clicking on “Endpoints” tab link in the Globus portal (after login) and click on the endpoint name. You should see “UUID” if you scroll down in the “Overview” section. Use UUID value for “fileContainerId”.

[Registering Data via the GUI](https://wiki.nci.nih.gov/x/aQmKFg)

[Register data object (Create & Update) via Command Line Utility](#_7.5_Register_a)

[Register a Directory via Command Line Utility](#_Register_a_Directory)

[Register data object (Create & Update) via Command Line Client](#_Registering_Data_Objects_1)

[Register data object (Create & Update) via CURL](#_Registering_a_data)

## Assign permissions on HPC DME collections or data objects

[Managing Permissions via the GUI](https://wiki.nci.nih.gov/x/s4mzFg)

[Sharing/Authorization via Command Line Client](#_Update_permissions)

[Sharing/Authorization via CURL](#_Assign_Permissions)

## Search functions

### [Find collection by metadata](#_Find_Project_by_1)

[Finding Data via the GUI](https://wiki.nci.nih.gov/x/pgOYFg)

[Find Collection by metadata via Command Line Client](#_Search_Collections_by)

[Find Collection by metadata via CURL](#_Search_for_collection)

### [Find data object by metadata](#_Find_Dataset_by_2)

[Finding Data via the GUI](https://wiki.nci.nih.gov/x/pgOYFg)

[Find Data Object by Metadata via Command Line Client](#_Search_Data_objects)

[Find Data Object by Metadata via CURL](#_Search_for_data)

The default hierarchical search API is based on the fact that every collection and dataObject inherits the metadata of all its parent collections. However, there should be no assumption made on the uniqueness of metadata attribute variables across a given hierarchy.

Assume a registered file in the following path: /Coll\_A/Coll\_B/Coll\_C/File\_D, it is possible that the same metadata variable “attribute\_X” be used for “Coll\_C” and “File\_D”. In this case, the value assigned to this attribute at the lower level (i.e., File\_D) will overwrite the value assigned to this attribute at the parent level (i.e., Coll\_C) unless otherwise specified as explained next.

Default hierarchical search using compound query (please review the [HPC S\_server\_ API Specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx)) will search the data object and collection at all levels (>=1) if no level value is indicated by the users. The lowest level in the hierarchy is given the value ‘1’. During the hierarchical search, users can also indicate the optional level they are basing the search on. For example, while searching for a dataObject, the dataObject level will equal ‘1’, the parent collection level will equal ‘2’, the grandparent collection level will equal ‘3’, etc.

Example 1: Given the path of File\_D shown above, assume you attached metadata attributes as the following:

* Coll\_A:  “X” = “1” (i.e., Metadata attribute name “X” set to a value of “1”)
* Coll\_B: “Y” = “2”
* Coll\_C: “Z” = “3”
* File\_D: “W” = “4”

Since these are unique metadata variables, there would be no difference made whether or not the users enter the level parameter values in entering the search criteria for the attribute “X”, “Y”, “Z”, or “W” because it will search at all levels.

Example 2: Assume you attached Metadata attributes as the following:

* Coll\_A: “X” = “1”
* Coll\_B: “Y” = “2”
* Coll\_C: “date\_created” = “11-11-2016”
* File\_D: “date\_created” = “11-20-2016”

Since “date created” may be assigned different values for the parent collection “Coll\_C” and the child “File\_D”, it is critical for the users to indicate at what level they are performing the search. However, if no level is indicated in the entered search criteria, default search will always search at all levels (collections and data files). If the user wants to search for a dataObject whose parent collection was created on a specific date, the compound query should include a search for the attribute “date\_created” with the value “11-11-2016” at level 2, with the level operator “EQUAL”. Therefore, the level operator specifies where to apply the search criteria.

Note that the default search level for data files is at all collection and data object levels. However, if a search is specifically indicated for collections, the default search is at all collection levels. In other words, while searching for collection, the search criteria will be executed at all collection levels (except data object level 1). The rationale is that there is nothing harmful to show a broader collections result set regardless if any data objects have been registered under certain collection path.

## Get collection by path

[Browsing for Data via the GUI](https://wiki.nci.nih.gov/x/rQOYFg)

[Get Collection Metadata by path via the Command Line Utility](#_Get_a_collection)

[Get Collection Metadata by path via Command Line Client](#_Search_Collection_by)

[Get Collection Metadata by path via CURL](#_Search_for_collection)

## Get data object by path

[Browsing for Data via the GUI](https://wiki.nci.nih.gov/x/rQOYFg)

[Find Data Object Metadata by Path via Command Line Utility](#_get_a_dataobject)

[Find Data Object Metadata by Path via Command Line Client](#_Search_Data_file)

[Find Data Object Metadata by Path via CURL](#_Search_for_data)

## Download functions

### Download data object/file to another globus endpoint

First setup your Globus account as mentioned [here](#_Using_Globus_for_1).

[Downloading Data via the GUI](https://wiki.nci.nih.gov/x/_QuKFg)

[Download data objects to Globus endpoint via CMD Line Utility](#_Download_a_Collection)

[Download data objects to Globus endpoint via CURL](#_Download_data_object)

### Download data object/file to local file system

[Downloading Data via the GUI](https://wiki.nci.nih.gov/x/_QuKFg)

[Download data objects to Local File System via CMD Line Utility](#_Download_a_dataobject)

[Download data objects to Local File System via CURL](#_Download_data_object_1)

# Executing HPC DME API with Command Line Client

The HPC DME command line client is an interactive tool where users can enter commands to initiate supported functions. Current version of HPC client supports the following functions:

* Batch registration of collections and their associated metadata
* Batch registration of data objects and their associated metadata
* Batch registration of local directories

Recursively register all files from a given local directory on your file system with HPC DME. This registers each directory and file with basic metadata and optional custom metadata. Inclusion and exclusion criterion can be set in include or exclude specific file and directory patterns.

* Batch registration of files from a Globus Endpoint

Recursively register all files from a given Globus endpoint location with HPC DME. This registers each file with basic metadata.

* Update user permissions on accessing HPC DME metadata
* Search for collections based on search criteria and write results into a csv/json/txt file
* Search for data objects based on search criteria and write results into a csv/json/txt file

HPC Client is integrated with HPC Server REST interfaces to perform the functions mentioned above. For batch registration of collection and dataObjects, it takes CSV (comma separated value) file as input for the commands it supports. HPC client parses given CSV file input and invokes server APIs in a batch manner to perform requested commands.

HPC Client commands are secured with valid authentication. To run protected commands, a user would need to provide HPC credentials to get authenticated. All client commands do not support multi-thread processing of input requests at this time.

## Seting the HPC DME CLI

Follow [these instructions](#_Get_the_HPC) to get the HPC DME CLI.

Before running a HPC client, you would need to update HPC client properties file /PATH/TO/HPC\_DME\_APIs/utils/hpcdme.properties

The following are the important properties:

|  |  |  |
| --- | --- | --- |
| Property name | Description | Default value |
| hpc.server.url | HPC Server API url. This is where REST interfaces are running on the server | Please see [Section 2.3](#_Accessing_HPC_DME) for environment specific API URL |
| hpc.ssl.keystore.path | Path of the keystore used for 2-way SSL connectivity with HPC Server. If the value is missing, one way SSL connection is used. | -For UAT server:  hpc-client/keystore/keystore-uat.jks  -For dev server:  hpc-client/keystore/keystore-dev.jks |

Please see [APPENDIX E](#_APPENDIX_E_HPC) for the full list of properties

## Running HPC DME Command line Client

HPC client is a command line tool that can be run using Java. Open cmd window and navigate to the folder where HPC client is extracted to. HPC Client can be run in interactive mode or batch mode.

### Interactive Mode:

Go to <HPC\_DME\_APIs>/utils folder and type the following command.

If you are using Unix:

export *HPC\_DM\_UTILS=<Path to HPC\_DM\_UTILS>*

*Java -Dhpc.client.properties=$HPC\_DM\_UTILS/hpcdme.properties -DHPC\_DM\_UTILS=$HPC\_DM\_UTILS -jar hpc-client/hpc-cli-<1>.<1>.<0>.jar*

If you are using Windows:

SET *HPC\_DM\_UTILS=<Path to HPC\_DM\_UTILS>*

*Java -Dhpc.client.properties=%HPC\_DM\_UTILS%/hpcdme.properties -DHPC\_DM\_UTILS=%HPC\_DM\_UTILS% -jar hpc-client/hpc-cli-<1>.<1>.<0>.jar*

You will see hpc-cli command line interface as below.



Type “help” to see all the commands supported by hpc-cli.



### Batch Mode

The HPC client supports running multiple HPC Client commands in a batch mode.

*Java -Dhpc.client.properties=$HPC\_DM\_UTILS/hpcdme.properties -DHPC\_DM\_UTILS=$HPC\_DM\_UTILS -jar hpc-client/hpc-cli-1.1.0.jar --cmdfile <Path to HPC commands>*

For example, you may put the following commands into a text file and give that file as --cmdfile input. HPC Client will run these commands in the sequence they are given in the file.

putCollections --source samples/Batch\_Upload\_Input\_File\_Collections.csv

putDatafiles --source samples/Batch\_Upload\_Input\_File\_Object.csv

putPermissions --source samples/Batch\_Permissions.csv

You could also run any operating system commands as part of the file by using “!”. When you are using batch mode, you should set “hpc.login.credentials” property value in hpc.properties. This is to bypass login prompt during batch execution. Please keep credentials file in a safe location so that only you have access to it.

#### Error Codes

HPC CLI returns error code if there is an error executing a given command. Following are the current error codes.

|  |  |
| --- | --- |
| Code | Category |
| CLI\_0 | Authentication error |
| CLI\_1 | Error reading file references from the properties files |
| CLI\_2 | Invalid user input |
| CLI\_3 | No input files to process |
| CLI\_4 | Failed to process collection |
| CLI\_5 | Failed to process data file |

#### Register collections

The HPC client supports registering multiple collections with HPC DME. A collection can be a project, sample, run, dataset or folder. New collection types can be added on HPC DME as needed by a system administrator as described above. The following is the command to register collections.

putCollections --source <input file path>

Please refer to <https://github.com/CBIIT/HPC_DME_APIs/tree/master/src/hpc-cli/samples> for sample input files.

Input file being used at the time of collections registration (mentioned in the command above) would need to provide all required metadata along with any user defined metadata. These attributes can be configured on HPC DME as needed by a system administrator.

Collections registration input file can have all collection types in a single file or they can be in separate files. If you are combining all collections into a single input file, attributes not relevant for a particular collection should have empty value.

Running the batch command would give an output as follows.



#### Registering Data Objects

The HPC client supports registering multiple data files with HPC DME. Parent collections need to be registered first before registering data files. The following is the command to register data files.

putDatafiles --source <input file path>

Input file being used at the time of data file or folder registration (mentioned in the command above) would need to provide all required metadata along with any user defined metadata. These attributes can be configured on HPC DME as needed by a system administrator. HPC DME supports registering data objects from a local file system (synchronously) or from a Globus endpoint (Asynchronously). Based on the given request in the input file, HPC DME client API initiates appropriate request.

Running the batch command would give an output as follows.



If the source of the data object(s) is a Globus endpoint, the expected operational behavior is that all data objects (files or folders) will be transferred via Globus Connect from the indicated “fileId” (associated with the “fileContainerId”) to the system configured Archive Area.

#### Bulk Registering of Collections and Data Objects from a Globus location

The HPC client supports registering all data files from a Globus location path. This is a convenient command to register all data files from a Globus location. This command recursively goes through each folder and registers all files along with folder structure. This process will generate “name” and “modified\_date” system metadata. This command expects that there is no data hierarchy nor metadata validation rules in place for the user DOC.

registerFromGlobusPath --globusEndpoint <Globus Endpoint UUID> --globusPath <Source Path on Globus> --destinationBasePath <Destination Base Path> -- includePatternFile <Pattern file to include source files> --excludePatternFile <Pattern file to exclude source files> --dryRun <true|false>

**globusEndpoint** : Source Globus endpoint

**globusSourcePath**: Source Path on the Globus endpoint. Please make sure to share this path with HPC DME Globus Group as detailed [Section 3.2 Using Globus for asynchronous](#_Globus_Account_for) data transfers.

**destinationArchivePath**: Destination path on the HPC DME archive

**dryRun**: Dry run to see list of source files after processing include and exclude patterns without actually registering the files with the archive. Valid values are “true” or “false”

**patternType**: Include, exclude criteria pattern type. Valid values are Simple, RegEx

**includePatternFile**: File name containing the list of patterns to match source files from registerFromFilePath to include. In the case of multiple patterns given in the input file, union of all patterns is considered.

**excludePatternFile**: File name containing the list of patterns to match source files from registerFromFilePath to exclude from included files. In the case of multiple patterns given in the input file, union of all patterns is considered.

**RegEx Pattern:**

With RegEx, the search pattern can be anything from a simple character, a fixed string or a complex expression containing special characters describing the pattern. The pattern defined by the regex may match one or several times or not at all for a given string. Please see the following link for patterns and examples.

<https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html>

**Simple Pattern**

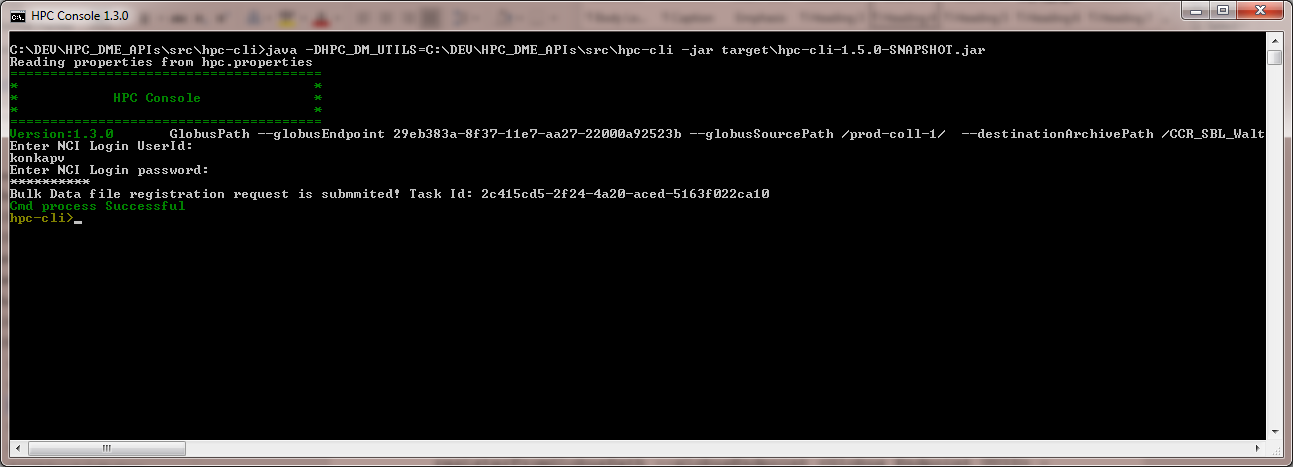
Pattern Matching:

|  |  |
| --- | --- |
| ? | Matches any single character. Eg, "something?" collects any path that is named "something" plus any character. |
| \* | Matches any characters up to the next slash. Eg, "\*/\*/something\*" collects any path that has two directories, then a file or directory that starts with the name "something". |
| \*\* | Matches any characters. Eg, "\*\*/something/\*\*" collects any path that contains a directory named "something". |

Ex:

|  |  |
| --- | --- |
| **\*.bam** | Matches all files with extension “bam” in the source folder |
| **FR00?.bam** | Matches all files with name “FR00” and any one character and with extension “bam” in the source folder |
| **\*\*/\*.bam** | Matches all files with extension “bam” in all folders and sub folders from the source location. |
| **images/\*\*/\*.jpg** | Matches all files with extension “jpg” in images folder and it’s all sub folders from the source location. |
| **\*\*/sub\*/\*.txt** | Matches all files in sub folder(s) starts with “sub” and with extension “txt” from the source location. |
| **ERI10wef** | Matches with exact file name from the source location |
| **/opt/test/temp.bam** | Matches with exact file name including its path from the source location |

Running the batch command would ask you to enter your NIH credentials to authenticate with HPC DME API. After successful submission, Task Id will be returned.



#### Bulk Registering of Collections and Data Objects from a local file path

The HPC client supports registering all data files from a local file path. This is a convenient command to register all data files or collections from a given path on your file system. This command recursively goes through each folder and register all files along with folder structure following given include and exclude patterns. This process will generate “name” and “modified\_date” metadata along with system generated metadata. This command expects that there is no data hierarchy and metadata validation rules in place for the user DOC. This command registers user provided metadata, if given along with the source files or folders. In order to register metadata, desired file should have corresponding <filename/folder name>.metadata.json file located in the same location as the source file or folder. This command parses the metadata JSON file and registers with along with the physical file or collection. If no metadata is given, this command registers basic metadata – Name, Created with the physical file or folder.

registerFromFilePath -- sourceFilePath<Local file path> --sourceFileList <source File list> -- includePatternFile <Pattern file to include source files> --excludePatternFile <Pattern file to exclude source files> -- filePathBaseName <Base path in the file path> --destinationArchivePath <Destination Base Path> --dryRun <true|false> --confirm <true|false> --metadata<true|false> --threads <number> --archiveType <S3|POSIX> --checksum <true|false>

**sourceFilePath**: Path on the local file system accessible to the CLI.

**sourceFileList:** File name containing the list of files with full path to register. This file list is relative to **sourceFilePath** value. If you are using sourceFileList**,** do not specifyincludePatternFile or excludePatternFile options

**destinationArchivePath**: Destination path on the HPC DME archive

**includePatternFile**: File name containing list of patterns to match source files from registerFromFilePath to include. The pattern can be set to match list of file extensions, like \*.bam. It can also be set to match part of file name. In the case of multiple patterns given in the input file, union of all patterns is considered.

**excludePatternFile**: File name containing the list of patterns to match source files from registerFromFilePath to exclude from included files. The pattern can be set to match list of file extensions, like \*.bam. It can also be set to match part of file name. In the case of multiple patterns given in the input file, union of all patterns is considered.

Pattern Matching:

|  |  |
| --- | --- |
| ? | Matches any single character. Eg, "something?" collects any path that is named "something" plus any character. |
| \* | Matches any characters up to the next slash. Eg, "\*/\*/something\*" collects any path that has two directories, then a file or directory that starts with the name "something". |
| \*\* | Matches any characters. Eg, "\*\*/something/\*\*" collects any path that contains a directory named "something". |

Ex:

|  |  |
| --- | --- |
| **\*.bam** | Matches all files with extension “bam” in the source folder |
| **FR00?.bam** | Matches all files with name “FR00” and any one character and with extension “bam” in the source folder |
| **\*\*/\*.bam** | Matches all files with extension “bam” in all folders and sub folders from the source location. |
| **images/\*\*/\*.jpg** | Matches all files with extension “jpg” in images folder and it’s all sub folders from the source location. |
| **\*\*/sub\*/\*.txt** | Matches all files in sub folder(s) starts with “sub” and with extension “txt” from the source location. |
| **ERI10wef** | Matches with exact file name from the source location |
| **/opt/test/temp.bam** | Matches with exact file name including its path from the source location |

**filePathBaseName**: This is the relative base path for your data registration. For example, if your files path is /opt/hpc/folder1/folder2/folder3 and you want your data file relative path to start with /folder1, set filePathBaseName as folder1. The logical file path will be destinationBasePath/folder1/folder2/folder3.

**dryRun**: Dryrun to see list of source files after processing include and exclude patterns without actually registering the files with the archive. Valid values are “true” or “false”

**confirm**: Ask user confirmation to register listing the folders/files. Valid values are “true” or “false”

**metadata**: Update metadata flag. If it is set to true, only metadata of the files with matching <file name/folder name>.metadata.json will be updated. No data files will be registered.

**threads**: Number of concurrent threads to run. In the case of multiple source files, each thread will process a file at a time. Make sure to set “upload.buffer.size” value in CLI properties suitable for you JVM settings so that your JVM will not run out of memory while processing large files. By default, this command runs with 1 thread with 1GB heap size on a 4GB memory system.

**archiveType:** Valid values are S3 or POSIX. If you specify “S3”, data is uploaded to S3 type archive using pre-signed URL. If you specify “POSIX”, data is uploaded through API Server and to the final POSIX file system.

**checksum:** By default, data file checksum is computed by the CLI and pass it to the API to verify checksum at the archival time. To turn it off, specify “false”.

Any errors during the process will be written to your HPC\_DME\_UTILS path. Any file that is not registered is written error records log. Running the batch command would ask you to enter your NIH credentials to authenticate with HPC DME API.



#### Update permissions

When a user registers a collection or data file with HPC DME, that user assumes “owner” role of the entity. Owner or system admin can assign permissions on these entities to other users or groups. HPC Client provides a convenient command to update permissions on multiple entities in a batch manner. The following is the command to assign permission on collections or data objects to a user or group.

putPermissions --source <input file path>

The following table shows default required attributes for input file. Please see [APPENDIX C](#Appendix c – sample permissions input) for sample batch input file.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| object\_path | Logical path of the entity (Collection or Data file) |
| UserId | HPC DME User Id to assign permission |
| GroupId | HPC DME Group Id to assign permission |
| Permission | Permission value.  Valid values are “write”, “own”, “read”, “none”  Own – Owner of the entity. Can assign permissions to other users  Read – Read only permission  Write – Read and Write permission  None – No permission |

Running the batch command would give an output as follows.



#### Search Collections by metadata

Using the following command, users can search for collections based on a given search criteria and write query results/records into a file.

getCollections --criteria '<Criteria file>' --format <csv|json|txt> --outputfile <file full path> --details <yes|no>

**Criteria**: HPC DME metadata search API criteria is a Json file with the search criteria. JSON file format should confirm to compound query format mentioned in [API Specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) Section 5.16 “Find Collection by compound metadata query”.

**format:** Search cmd takes optional “format” argument to specify format of the output file. Valid options are “csv”, “json” and “txt”. If this argument is not given, output is written in “txt” format. For “csv” format, collections “selfMetadataEntries” are written to the output file, “parentMetadataEntries” will not be written into the output file.

**outputfile**: Search cmd takes optional “outputfile” argument to specify output file full path. If path is not given, default path is “hpc.error-log.dir” + “getcollections\_records”+<Current date> + <type>.

**details**: Search cmd takes optional “details” argument to specify if output should include collection details or not. If not specified, default value is “yes”. If “no” value is given, output file written with collection path values only.

#### Search Collection by path

Using the following command, users can search for collections based on a logical path and write resulted search records into a file.

getCollection --path '<path>' --format <csv|json|txt> --outputfile <file full path>

**path:** Logical path of the collection

**format:** As explained in the section above

**outputfile:** As explained in the section above

#### Search Data objects

Using this command, users can search for data files based on a given search criteria and write retrieved search results/records into a file.

getDatafiles --criteria '<Criteria>' --format <csv|json|txt> --outputfile <file full path> --details <yes|no>

**Criteria**: HPC DME metadata search API criteria is a json file with the search criteria. JSON file format should confirm to compound query format mentioned in [API Specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) Section 5.25 for Data files.

**format:** Search cmd takes optional “format” argument to specify format of the output file. Valid options are “csv”, “json” and “txt”. If this argument is not given, output is written in “txt” format. For “csv” format, collections “selfMetadataEntries” are written to the output file, “parentMetadataEntries” will not be written into the output file.

**outputfile**: Search cmd takes optional “outputfile” argument to specify output file full path. If path is not given, default path is “hpc.error-log.dir” + “getDatafiles\_records”+<Current date> + <type>.

**details**: Search cmd takes optional “details” argument to specify if output should include collection details or not. If not specified, default value is “yes”. If “no” value is given, output file written with data files path values only.

#### Search Data file by path

Using the following command, users can search for data file based on a given logical path and write retrieved search results/record into a file.

getDatafile --path '<path>' --format <csv|json|txt> --outputfile <file full path>

**path:** Logical path of the data file

**format:** As explained in the section above

**outputfile:** As explained in the section above

#### Delete Data Object

Using the following command, users can delete a data file from the archive.

deleteDatafile --path '<file\_path>'

**path**: Location of the data object to be deleted in the archive

**Example**:

deleteDatafile --path /TEST\_NO\_HIER\_Archive/downloads/testfile10

#### Delete Collections

Using the following command users can delete an empty collection or delete a collection recursively.

deleteCollection –path <collection\_path> --recursive <true|false>

**path**: Location of the collection to be deleted in the archive.

**recursive**: Delete the data objects and sub-collections within the collection recursively. If set to false, the collection is deleted only if empty. If not specified, recursive defaults to false.

**Example:**

deleteCollection --path /TEST\_NO\_HIER\_Archive/downloads --recursive true

#### Enroll User

This command creates a new user in the system. It can only be executed by the system admin. or the group admin.

dm\_register\_user <userId> <description.json>

**userId**: The user Id of the NIH AD account of the user to be added to the system.

**<description.json>:** The name of the JSON file conatining the data params for the command. For the contents of the <description.json> file, please refer to the “Enroll User” in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for details.

#### Update User

This command updates a user already enrolled in the system. It can only be executed by the system admin. or the group admin.

dm\_update\_user <userId> <description.json>

**userId**: The user Id of the NIH AD account of the user to be updated in the system.

**<description.json>:** The name of the JSON file conatining the data params for the command. For the contents of the <description.json> file, please refer to the “Update User” in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for details.

#### Add Bookmark

This command allows the user to add a new bookmark. It also allows the system admin or group admin to add bookmarks for other users.

dm\_add\_bookmark <bookmark\_name> <description.json>

**<bookmark\_name>**: The name that will be displayed on the UI for the bookmark.

**<description.json>:** The name of the JSON file containing the data params for the

command. For the contents of the <description.json> file, please refer to the “Enroll

User” in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for details.

## Error handling

HPC client interface catches any errors returned by HPC Server API and write them into error log file. The error log is written as <record sequence number> <Error message>.

Those records not processed by HPC client due to any error are written into a separate file “errorRecords<timestamp>.txt” for each command so that only these records can be corrected to go through the HPC batch client again.

# Executing HPC DME API with Command line utilities

Some of the curl command mentioned in the [curl section](file:///C:\Users\luz6\AppData\Local\Temp\HPC_User_Guide-1.docx#_Acessing_HPC_DME) and the batch registration of direcotires mentioned in [HPC DME CLI section](#_Batch_Mode) have been wrapped with bash functions to make them easily accessible from command line.

These functions are available as part of the HPC DM API git repository:

<https://github.com/CBIIT/HPC_DME_APIs/tree/master/utils>

Here are examples on how to use some of the utilities:

## One time setup

Follow [the instructions](#_Pre-Requisites_for_HPC) to get the latest version of the utilities.

Follow the /path/to/HPC\_DME\_APIs/utils/README.md file for the one time setup instructions of the utilities. All utilities start with the prefix “dm\_”.

## Generate a Data management API token

Run the following command to generate a DM API token (instead of putting your password every time):

dm\_generate\_token

## Generate a Globus token

If you would like to register directories directly from Globus, first you need to edit your Globus information as mentioned in the /path/to/HPC\_DME\_APIs/utils/README.md file. Then run the following command to generate the Globus token:

dm\_globus\_generate\_token

Note that you need to create a Globus account and link it to your NIH account as explained [here](#_using_Globus_for).

## Register a collection

To register a collection, use the command:

dm\_register\_collection <collection-description.json> <destination-path>

The git repository contains a [sample file for collection-description.json](https://github.com/CBIIT/HPC_DME_APIs/blob/master/utils/templates/collection-metadata.json). Feel free to edit the sample file to add more metadata to your collection.

For example, your command can be:

dm\_register\_collection /path/to/HPC\_DME\_APIs/utils/templates/collection-metadata.json /<top-level-directory>/my-collection

## Register a data object

### Synchronously (From file system)

To register a data object synchronously, use the command:

dm\_register\_dataobject <dataobject-description.json> <destination-path> <source-file>

The git repository contains [a sample file for dataobject-description.json](https://github.com/CBIIT/HPC_DME_APIs/blob/master/utils/templates/dataobject-sync-metadata.json).

For example, your command can be (after registering a collection above):

$dm\_register\_dataobject  /path/to/HPC\_DME\_APIs/utils/templates/dataobject-sync-metadata.json /<collection-path>/my-dataobject /path/to/dataobject-file

### Asynchronously (From Globus)

To register a data object asynchronously, use the command:

dm\_register\_dataobject <dataobject-description.json> <destination-path>

The dataobject-descripiton file needs to contain a valid Globus endpoint and path for your source dataObject. Also, the corresponding HPC DME Globus Group account must have a read permission to the Globus endpoint as indicated ­Using [Using Globus for asynchronous data transfers](#_Globus_Account_for). The repository contains [a sample file for asynchronous, dataobject-description.json,](https://github.com/CBIIT/HPC_DME_APIs/blob/master/utils/templates/dataobject-async-metadata.json) for asynchronous registration of a data object as shown in this example:

dm\_register\_dataobject  /path/to/HPC\_DME\_APIs/utils/templates/dataobject-async-metadata.json /<collection-path>/my-dataobject

## Register a Directory

With the command line utilities you can recursively register all files within a directory and its subdirectories to the archive. By running the two commands listed below, basic metadata information (e.g., filename, size, etc.) will be created for every file and the file will be pushed to the archive. The user is expected to append any otherre available metadata attributes- at later time.

### Synchronously (from file system)

To register a directory from your file system, execute the command:  
  
dm\_register\_directory [OPTIONS] <source-directory-path-on-file-system> <destination-path-on-archive>

OPTIONS:

-d                    Print the files to be included in registration an exit

-i[INCLUDE\_FILE]      Only include the files that match the patterns in INCLUDE\_FILE

 -e[EXCLUDE\_FILE]    Exclude the files that match the patterns specified in EXCLUDE\_FILE

  -h                    Print this message

This command will register all files and directories from the source path to the Archive’s destination path. Regular expressions patterns can be used through the include/exclude options to register only a subset of the files in the source directory. File names that match any of the include regular expression patterns are matched first, then the names that match any of the exclude patterns are removed from the registration job. For more information about the how to write the regular expression patterns, check [this section](#_Bulk_Registering_of).

Metadata can be automatically added for every file that will be registered. The metadata file should have the name <original-file-name>.metadata.json. The format of the metadata file should be similar to the one defined in the dataObject registration API. Check the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for more information.

### Asynchronously (from Globus)

To register a directory that exists on a Globus endpoint, first generate a Globus token as shown [here](#_Generate_a_Globus), then execute the command:  
  
dm\_register\_globus\_directory <source-directory-path-globus-endpoint> <destination-path-on-archive> [globus-endpoint]

Note that by default, the Globus endpoint you defined in your hpcdme.properties file is used as mentioned in the README.md. If you would like to use a different Globus endpoint, you can add the optional endpoint UUID at the end of the command.

## Get a collection metadata

To get all the metadata associated with a collection, execute the command:

dm\_get\_collection <collection-path> [response-message] [response-header]

If the [response-message] is omitted, the metadata will be printed to stdout.

## Get a dataobject metadata

To get all the metadata associated with a dataObject, execute the command:

dm\_get\_dataobject <dataobject-path> [response-message] [response-header]

If the [response-message] is omitted, the metadata will be printed to stdout.

## Download a dataobject synchronously

To download a dataObject file from the archive to your current file system, execute the command:

dm\_download\_dataobject\_sync <dataobject-path-on-arhive> <destination-path-on-local-file-system>

## Download a Collection or dataobject Asynchronously

To download a dataObject or a collection from the archive to a Globus endpoint, execute the command:

dm\_download\_async <source-path-on-arhive> <destination-path-on-globus-endpoint> [globus-endpoint]

Note that by default, the Globus endpoint you defined in your hpcdme.properties file is used as mentioned in the README.md. If you would like to use a different Globus endpoint, you can add the optional endpoint UUID at the end of the command.

## Download a Collection or DataObject to an s3 bUCKET

To download a dataObject or a collection from the archive to an S3 bucket, execute the command:

dm\_download\_s3 <source-path-on-archive> <s3\_description.json>

where the s3\_description.json file provides the destination location and S3 credential information in the following format:

{

"s3DownloadDestination": {

"destinationLocation": {

"fileContainerId": "<name of the bucket to download to>",

"fileId": "<file or folder name in the bucket>"

},

"account": {

"accessKey": "<access key ID for the account>",

"secretKey": "<secret access key for the account>",

"region": "<s3 region e.g. us-east-2>"

}

}

}

## Delete a Dataobject From the Archive

dm\_delete\_datafile <file\_path\_on\_archive>

**Example:**

dm\_delete\_datafile /TEST\_NO\_HIER\_Archive/downloads/testfile10

## Delete a Collection from the Archive

To delete an empty collection or to delete a collection recursively from the Archive, execute the command:

dm\_delete\_collection [OPTIONS] <collection-path\_on\_archive>

**OPTIONS:**

  -r Delete all files and sub-collections under the specified path recursively.

If the –r option is not specified, the collection will be deleted only if it is not empty.

**Example:**

dm\_delete\_collection –r /TEST\_NO\_HIER\_Archive/downloads

## Move/Rename a file or collection in the Archive

To move a collection or file from one location to another in the Archive, execute the command:

dm\_rename <move-description.json>

where the move-description.json file provides the list of source and destination locations in the following format:

{

"moveRequests": [

{

"sourcePath": <full path in the Archive to the file/collection to be moved>

"destinationPath": <full path in the Archive to the new file/collection>

}

]

}

Note that the destination collection or file should not be existing prior to executing this command.

**Example:**

Below is an example command to move a collection, and a file:

dm\_rename “my-move-list.json”

where my-move-list.json contains:

{

"moveRequests": [

{

"sourcePath": "/TEST\_Archive/PI\_Lab1/Project\_Orig",

"destinationPath": "/TEST\_Archive/PI\_Lab2/Project\_New"

},

{

"sourcePath": "/TEST\_Archive/PI\_Lab1/Project\_Name1/File\_Orig",

"destinationPath": "/TEST\_Archive/PI\_Lab2/Project\_Name2/File\_New"

}

]

}

# Executing HPC DME API with CURL

Curl is a command-line tool for transferring data using various protocols. It can be used to interact with the HPC DME REST API. If you are using Windows environment, you will need to install tools like CygWin to be able to run Unix-like commands. Please visit <https://www.cygwin.com/> to download. Here are examples on how to use the curl tool to submit REST requests to the HPC DME server.

## Setting up the environment

First make sure you have curl in your environment path. Open a terminal and type:  
$curl

You should get the following line printed on stdout:

curl: try 'curl --help' or 'curl --manual' for more information

If you do not have curl on your path, check your Linux distribution to install it.

## Executing CURL commands

In the examples shown below, any string between the two characters ‘<’ and ‘>’ is a mandatory string that should be replace with an actual value without the enclosing characters ‘<’ and ‘>’.   
  
For example, <server:port> should be replaced with the HPC DME server name and port number (e.g., <https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738)>

In these examples, the returned response header is dumped to a file using the -D optional url parameter. For example -D <curl-response-header-file>. Similarly, I dump the response message as a json file using the -o flag. For example: -o <curl-response-message.json>

### Generate Token

curl -k -H "Content-Type: application/json" -X GET <server>/authenticate -H "Accept: application/json" -D AuthenticateGet-header1.txt -o Response\_Token.txt -u sehgalu2

**Example – Generate token**

curl -k -H "Content-Type: application/json" -X GET https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/authenticate -H "Accept: application/json" -D header.txt -o Response\_token.txt -u sehgalu2

### Generating a token will help you authenticate in all your subsequent commands. This will eliminate the need to enter the NIH password everytime you run a curl command. Once you run the Authenticate API, a token will be generated and stored in the response file (Response\_Token.txt as shown in the example above). You must copy the generated token to a new file – curl-conf (you have to create this file) and save that file in your current working directory.

### Content structure of the *curl-conf* file. Please replace the token shown below with your token. The token starts with “ey’

### -k

### -H "Authorization : Bearer eyJhbGciOiJIUzI1NiJ9..EvEVKsZaUkg-zxmXdIIMNipLE8C-ftRKuygaFfHt1LU"

Once you have the **curl-conf** file saved with your generated token, you can now use the file in all of your subsequent curl commands to authenticate. Example below shows a bookmark being added.We are authenticating using our curl-conf file which has our newly generated token.

**Example – Using curl-conf file in a CURL command** **to authenticate**

curl --config tokens/curl-conf -k -H "Content-Type: application/json" -d @/Users/sehgalu2/HPC\_DME\_APIs\_1.18.0/utils/uditbookmark.json -X PUT https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/bookmark/18thsept\_test -H "Accept: application/json" -D demo1\_udit.txt -o output\_1\_udit.txt

### Registering a new user

curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X PUT <server>/user/<NCI-USER-Id> -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

**Example - Attribute-File.json**

{{

"firstName": "John",

"lastName": "Doe",

“defaultBasePath”: “/TEST\_NO\_HIER\_Archive”,

"userRole": “USER”

}}

**Example to Enroll user**

curl -k -H "Content-Type: application/json" -d @userinfo.json -X PUT https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/user/varadarajans -H "Accept: application/json" -D demo1\_udit.txt -o output\_1\_udit.json -u sehgalu2

Check out “Enroll User” description in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for details.

### Registering a collection

curl -H "Content-Type: application/json" -d @<attributes-file.json> -X PUT <server>/collection/<collection-path> -H "Accept: application/json" -D <response-header> -o <response-message.json>   
  
The “attributes-file.json” should include all the mandatory attributes associated with a collection type.

### Registering a data object

As mentioned in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx), there are two methods to register data objects: asynchronously using Globus, or synchronously from the file system.

#### 8.2.3.1 Registering a data object from a Globus endpoint

curl -F dataObjectRegistration=@<attributes-file.json>;type=application/json -X PUT <server>/dataObject/<dataObject-path> -H "Accept: application/json" -D <response-header> -o <response-message.json>  
  
Note that the “attributes-file.json” should include the “fileContainerID” and the“fileId” as mentioned in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx).

#### 8.2.3.2 Registering a data object from the file system

curl -F "dataObjectRegistration=@<attributes-file.json>;type=application/json" -F "dataObject=@<dataObject-file>;type=application/octet-stream" -X PUT <server:port>/ /dataObject/<dataObject-path>

Note that the attributes-file.json should NOT include the “fileContainerID” nor the “fileId” as mentioned in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx).

#### 8.2.3.3 Registering data object to a new parent collection

### curl -k -F "dataObjectRegistration=@<attributes-file.json>;type=application/json" -F "dataObject=@<dataObject-file>;type=application/octet-stream" -X PUT <server:port>/ dataObject/<dataObject-path> -D <response-header> -o <response-message.json> -u <YourNCI User id>

Note: The ‘createParentCollections’ indicator needs to be set to true, and a list of metadata to create the parent collections provided through parentCollectionBulkMetadataEntires. In this case the entire collection hierarchy above the registered collection will be created

**Example – Attributes-file.json**

### {

### "metadataEntries": [{

### "attribute": "object\_name",

### "value": "demo\_1\_udit"

### },

### {

### "attribute": "collection\_type",

### "value": "udit\_1\_project"

### }

### ],

### "createParentCollections": true,

### "parentCollectionsBulkMetadataEntries": {

### "defaultCollectionMetadataEntries": [{

### "attribute": "collection\_type",

### "value": "Folder"

### }]

### }

### }

### Example – Uploading Data File to a new parent collection

### curl -k -F "dataObjectRegistration=@Attributes-File.json;type=application/json" -F "dataObject=@/Users/sehgalu2/Documents/GitHub/HPC\_DME\_APIs/utils/do1.json;type=application/octet-stream" -X PUT https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/dataObject/TEST\_NO\_HIER\_Archive/udit\_trial\_2/do1 -D test\_run8\_us.txt -o perm\_test8\_udit.json -u sehgalu2

### Search for collection

Curl -X POST <server>/collection/query/compound –H "Content-Type: application/json" -d @<compound-query.json> -H "Accept: application/json" -D <response-header> -o <response-message.json>

The “compound-query.json” file needs to contain the attribute to be searched, and the comparison operation is indicated in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx).

### Search for data Object

Curl -X POST <server>/dataObject/query/compound –H "Content-Type: application/json" -d @<compound-query.json> -H "Accept: application/json" -D <response-header> -o <response-message.json>

The “compound-query.json” file needs to contain the attribute to be searched, and the comparison operation is indicated in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx).

### Search for a list of objects in a collection

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2/dataObject/query/path-to-collection -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{

"compoundQuery": {

"operator": "AND",

"queries": [{

"attribute": "registered\_by",

"value": "%sehgalu2%",

"operator": "LIKE"

},

{

"attribute": "data\_transfer\_completed",

"value": "04-05-2019 14:45:23",

"format": "MM-DD-YYYY HH24:MI:SS",

"operator": "TIMESTAMP\_GREATER\_OR\_EQUAL"

}

]

},

"detailedResponse": true,

"page": 1,

"totalCount": true

}}

### Example – Search for a list of data objects in a collection

curl -k -H "Content-Type: application/json" -d @Attributes-File.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/dataObject/query/TEST\_NO\_HIER\_Archive/udit\_1\_collection -H "Accept: application/json" -D demo1\_udit.txt -o output\_1\_udit.json -u sehgalu2

### Assign Permissions

curl -H "Content-Type: application/json" -d @<permission-file.json> -X POST <server>/acl -H "Accept: application/json" -D <response-header> -o <response-message.json>

Check out “Set Collection Permissions” in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for details.

### Download data object to a Globus endpoint

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2/dataObject/path-to-data-file/download -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{

"globusDownloadDestination": {

"destinationOverwrite" : true,

"destinationLocation": {

"fileContainerId": "16572124-19cb-11e9-934d-0e3d676669f4",

"fileId": "globus\_file\_2"

}

}

}

### Example – Download data file from DME to Globus

curl -k -H "Content-Type: application/json" -d @Attributes-file.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/v2/dataObject/TEST\_NO\_HIER\_Archive/udit\_move3\_collection/dataobject1.json/download -H "Accept: application/json" -D arch\_local -o arch\_to\_local.json -u sehgalu2

Check out description and details on how to Download file to globus on [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx).

### Download data object to a file system (deprecated)

curl -H "Content-Type: application/json" -d @<empty.json> -X POST <server>/dataObject/<object-path> /download -H "Accept: application/json" -D <response-header> -o <response-message>  
  
In this call, the “empty.json” file should contain just the two characters “{}”. The data object will be downloaded with the name “response-message”

### Download data object to a file system(V2)

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2/dataObject/path-to-data-file/download -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{}

### Example to download data object to a file system(V2)

curl -k -H "Content-Type: application/json" -d @Attributes-File.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/v2/dataObject/TEST\_NO\_HIER\_Archive/udit\_move3\_collection/dataobject1.json/download -H "Accept: application/json" -D arch\_local -o arch\_to\_local.json -u sehgalu2

Check out description and details on how to download Data boject from DME to Local drive on [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx).

### Download data object to an S3 bucket

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2/dataObject/path-to-data-file/download -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{

"s3DownloadDestination": {

"destinationLocation": {

"fileContainerId": "cricketics",

"fileId": "demo\_udit\_collection"

},

"account": {

"accessKey": "AK3AIVTU73NOEDJVBGQA",

"secretKey": "OWdZXKPI4U1aaFTgoEUD8SEx2+2S0SWulxvnPc9Q",

"region": "us-east-1"

}

}

}

### Example to download data object to S3 bucket

curl -k -H "Content-Type: application/json" -d @Attributes-File.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/v2/dataObject/TEST\_NO\_HIER\_Archive/udit\_move3\_collection/dataobject1.json/download -H "Accept: application/json" -D s3\_local.txt -o s3\_to\_local.json -u sehgalu2

Check out description and details on how to download data file to Amazon on S3 on [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx).

### Download collection to an S3 bucket

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2/collection/path-to-data-file/download -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{

"s3DownloadDestination": {

"destinationLocation": {

"fileContainerId": "cricketics",

"fileId": "demo\_udit\_collection"

},

"account": {

"accessKey": "AK3AIVTU73NOEDJVBGQA",

"secretKey": "OWdZXKPI4U1aaFTgoEUD8SEx2+2S0SWulxvnPc9Q",

"region": "us-east-1"

}

}

}

### Example – Download collection to an S3 bucket

### curl -k -H "Content-Type: application/json" -d @Attributes-file.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/v2/collection/TEST\_NO\_HIER\_Archive/udit\_move3\_collection/download -H "Accept: application/json" -D s3\_local.txt -o s3\_to\_local.json -u sehgalu2

Check out description of this API and attribute-file.json example format in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for details.

### Delete Data Object (by group administrators only)

A group administrator is suggested to use the [Web GUI](#_Delete_a_Data) to delete a data file registered by error, which is covered in the [Group Administration section](#_Delete_a_Data).

### Move a collection

curl -u <user-id> -k -X POST <server>/collection/<source-path>/move/<destination-path>

<user-id> - The NIH user-id

<server> - The server endpoint

<source-path> - The move source path. This must be url-encoded string

<destination-path> - The move destination path. This must be url-encoded string

### Move a data object

curl -u <user-id> -k -X POST <server>/dataObject/<source-path>/move/<destination-path>

<user-id> - The NIH user-id

<server> - The server endpoint

<source-path> - The move source path. This must be url-encoded string

<destination-path> - The move destination path. This must be url-encoded string

### Bulk move

curl -u <user-id> -H "Content-Type: application/json" -k -X POST <server> /move -d ‘<json>’

<user-id> - The NIH user-id

<json> -

{

"moveRequests": [

{

"sourcePath" : "/FNL\_SF\_Archive/eran-pi-lab/eran-project/eran-flowcell/eran-sample-3",

"destinationPath" : "/FNL\_SF\_Archive/eran-pi-lab/eran-project/eran-flowcell/test/eran-sample-4"

}

]

}

### Update User

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/user/{nciUserId} -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attribute-File.json

### {

### "firstName": "Udit",

### "lastName": "Sehgal",

### “defaultBasePath”: “/TEST\_Archive”,

### "userRole": “System\_Admin”,

### “active”: "true"

### }

### 

### Example to Update User

curl -k -H "Content-Type: application/json" -d @ Attribute-File.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/user/sehgalu2 -H "Accept: application/json" -D demo1\_udit.txt -o output\_1\_udit.json -u sehgalu2

Check out “Update User” description in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for details.

**Note**: Only System Administrator can update any user**.**

### Add a group

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X PUT <server>/group/{groupName} -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attribute-File.json

{

"addUserIds": ["sehgalu2","menons2"]

}

**Example to add a group**

curl -k -H "Content-Type: application/json" -d @Attribute-File.json -X PUT https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/group/jantest -H "Accept: application/json" -D demo1\_udit.txt -o output\_1\_udit.json -u sehgalu2

Check out “Add a group” description in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for details

### Add a bookmark

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X PUT <server>//bookmark/{bookmark name} -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <Your NCI User id>

### Example - Attribute-File.json

{

"path": "/TEST\_NO\_HIER\_Archive/udit\_nci",

"group": "fridaytestgroup",

"userId": "sehgalu2",

"permission": "WRITE"

}

### Important note: The fields 'userId' and 'permission' are available only for system admin. and group admin. to set bookmark and permission for other users. If no permission is specified, then the existing permission is left as is.

### Example of how to Add a Bookmark

### curl -k -H "Content-Type: application/json" -d @Attribute-File.json -X PUT https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/bookmark/fridaybookmark -H "Accept: application/json" -D demo1\_udit.txt -o output\_1\_udit.json -u sehgalu2

Check out “Add a Bookmark” description in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for details

### Update bookmark

### curl -k -H "Contet-Type: application/json" -d @<attributes-file.json> -X PUT <server>/bookmark/{bookmark name} -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <Your NCI User id>

### Example - Attribute-File.json

{

"path": "/TEST\_NO\_HIER\_Archive/udit\_nci ",

"group": "testgroup",

"userId": "sehgalu2",

"permission": "WRITE"

}

### Example of how to Update a bookmark

### curl -k -H "Content-Type: application/json" -d @Attribute-File.json -X PUT https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/bookmark/fridaybookmark -H "Accept: application/json" -D demo1\_udit.txt -o output\_1\_udit.json -u sehgalu2

### Important note: The fields 'userId' and 'permission' are available only for system admin. and group admin. to set bookmark and permission for

### other users

Check out “Update a Bookmark” description in the [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx) for details

### Download Collection from DME TO GLOBUS

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2collection/<collection-path>;/download -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{

"globusDownloadDestination": {

"destinationOverwrite": true,

"destinationLocation": {

"fileContainerId": "16DDF572124-19cb-11e9-934d-0e3d676669f4",

"fileId": "globus\_file\_2"

}

}

}

### Example – Download data file from DME to Globus

### curl -k -H "Content-Type: application/json" -d @globus\_upload.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/v2/collection/TEST\_NO\_HIER\_Archive/udit\_move3\_collection/download -H "Accept: application/json" -D s3\_local.txt -o s3\_to\_local.json -u sehgalu2

Check out description and details on how to download collection to Globus on [HPCDME API specification](https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/guides/HPC_API_Specification.docx).

8.2.21.1 **Get Collection Information about a Specfic Collection**

curl -k <server>/collection/{collectionPath}?list=[true|false] -D <response-header> -o <response-message.json> -u <YourNCI User id>

**Example – Get Collection Information for a Specific Collection**

curl -k 'https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/collection/FNL\_SF\_Archive?list=true' -D header.txt -o output.txt -u sehgalu2

### 8.2.21.2 Get Collection Information along with user permission for a specific collection

### curl -k <server>/collection/{collectionPath}?list=[true|false]&includeAcl=[true/false]' -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Get Collection Information with user permission for a specific collection

curl -k 'https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/collection/FNL\_SF\_Archive?list=true&includeAcl=true' -D header.txt -o output.txt -u sehgalu2

### 8.2.22.1 Get Data Management Model for all basepaths

### curl -k -X GET <server>/dm/model -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example – Data Management model for all basepaths

### curl -k -X GET https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/dm/model -u sehgalu2

### 8.2.22.2 Get Data Management Model for a specific basepath

### curl -k -X GET <server>/dm/model/basepath -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example – Data Management model for a specifc basepath

### curl -k -X GET https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/dm/model/FNL\_SF\_Archive -u sehgalu2

### 8.2.23.1 Download a specific file from an archived tar

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2/dataObject/path-to-data-file/download -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{

"synchronousDownloadFilter": {

"compressedArchiveType": "TAR",

"includePatterns": ["\*\*/pi-xyz\*.\*", "LDT\_INTRO/Post\_Reflection.json"]

}

}

### Example – Download a particular file from an archived tar

curl -k -H “Content-Type: application/json” -d @/Users/sehgalu2/HPC\_DME\_APIs\_1.17.0/utils/Attributes-File.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/v2/dataObject/FNL\_SF\_Archive/pilabus/uditproject/uditflowcell/udittest/archive2.tar/download-H “Accept: application/json” -D arch\_local\_header.txt -o arch\_to\_local.tar -u sehgalu2

### 8.2.23.2 Download two files from from an archived tar

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2/dataObject/path-to-data-file/download -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{

"synchronousDownloadFilter": {

"compressedArchiveType": "ZIP",

"includePatterns": ["\*\*/pi-xyz\*.\*", "Tarred/3gbfile.docx", “Tarred/4gbfile.docx”]

}

}

### Example – Download a particular file from an archived tar

curl -k -H "Content-Type: application/json" -d @/Users/sehgalu2/HPC\_DME\_APIs\_1.17.0/utils/Attributes-File.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/v2/dataObject/FNL\_SF\_Archive/udittest4/Uditproject/FlowcellUdit/sampleudit/Tarred2.tar/download -H "Accept: application/json" -D arch\_local.txt -o outputudit2.tar -u sehgalu2

### 8.2.23.3 Download all files with a file name pattern (example – download all files with “.txt” in its name)

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2/dataObject/path-to-data-file/download -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{

"synchronousDownloadFilter": {

"compressedArchiveType": "ZIP",

"includePatterns": ["\*\*/\*.txt\*"]

}

}

### Example – Download files with pattern “.txt”

curl -k -H "Content-Type: application/json" -d @/Users/sehgalu2/HPC\_DME\_APIs\_1.17.0/utils/Attributes-File.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/v2/dataObject/FNL\_SF\_Archive/udittest4/Uditproject/FlowcellUdit/sampleudit/Tarred3.tar/download -H "Accept: application/json" -D arch\_local.txt -o outputudit3.tar -u sehgalu2

### 8.2.24.1 Get data object information

### curl -k <server>/dataObject/{dataObject-Path}?list=[true|false]' -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Get data object information

curl -k -u sehgalu2 'https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/dataObject/FNL\_SF\_Archive/udittest4/Uditproject/FlowcellUdit/sampleudit/Tarred2.zip?list=true'

### 8.2.24.2 Get Data Object informaton with the user permission

### curl -k <server>/dataObject/{dataObject-Path}?list=[true|false]&includeAcl=[true/false]' -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Get data object Information with the user permission

curl -k -u sehgalu2 'https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/dataObject/FNL\_SF\_Archive/udittest4/Uditproject/FlowcellUdit/sampleudit/Tarred2.zip?list=true&includeAcl=true'

**8.2.25 Download list of Collections to Globus**

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2/ download -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{

"collectionPaths" : [

"/FNL\_SF\_Archive/udittest\_collection\_1",

"/FNL\_SF\_Archive/udittest\_collection\_2"

],

"globusDownloadDestination": {

"destinationOverwrite" : true,

"destinationLocation": {

"fileContainerId": "16572124-19cb-11e9-934d-0e3d676669f4",

"fileId": "bulk-collections-v2-09-9-05-19"

}

}

}

### Example – Download list of collections

### curl -k -H "Content-Type: application/json" -d @/Users/sehgalu2/HPC\_DME\_APIs\_1.17.0/utils/Attributes-File.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/v2/download -H "Accept: application/json" -D GLOBUS.txt -o collections.txt -u sehgalu2

**8.2.25.2 Download a list of Collections to AWS S3**

### curl -k -H "Content-Type: application/json" -d @<attributes-file.json> -X POST <server>/v2/ download -H "Accept: application/json" -D <response-header> -o <response-message.json> -u <YourNCI User id>

### Example - Attributes-File.json

{

"collectionPaths" : [

"/FNL\_SF\_Archive/udittest",

"/FNL\_SF\_Archive/udittest2"

],

"s3DownloadDestination": {

"destinationLocation": {

"fileContainerId": "cricketics",

"fileId": "9\_13\_test"

},

"account" : {

"accessKey" : "AKIAIVesTU7ANOEDJVBGQA",

"secretKey" : "OWdZXKPI4U1uuEEFTgoEUD8SEx2+2S0SWulxvnPc9Q",

"region" :"us-east-1"

}

}}

### Example – Download list of collections to AWS S3

### curl -k -H "Content-Type: application/json" -d @/Users/sehgalu2/HPC\_DME\_APIs\_1.17.0/utils/Attributes-File.json -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/v2/download -H "Accept: application/json" -D s3\_local.txt -o collections.txt -u sehgalu2

### 

**8.2.25.2 Generate a Download Request URL**

### curl -k -H "Content-Type: application/json" -X POST <server>/dataObject/<dataObjectPath>/generateDownloadRequestURL -H "Accept: application/json" -D <response-header> -o <response-message.txt> -u <YourNCI User id>

### Example – Generate a Download Request URL

curl -k -H "Content-Type: application/json" -X POST https://fr-s-hpcdm-uat-p.ncifcrf.gov:7738/hpc-server/dataObject/FNL\_SF\_Archive/udittest4/Uditproject/FlowcellUdit/sampleudit/9\_12\_archive.tar/generateDownloadRequestURL -H "Accept: application/json" -D header.txt -o response-message.txt -u sehgalu2

### 

# Executing HPC DME API with Web Client

The following wiki space describes how to execute HPC DME API with the web client:

<https://wiki.nci.nih.gov/display/DMEdoc>



# APPENDIX A – sample collection input

[https://github.com/CBIIT/HPC\_DME\_APIs/tree/master/utils/hpc-client/samples/Batch\_Upload\_Input\_File\_Collections.csv](https://ncisvn.nci.nih.gov/svn/HPC_Data_Management/branches/hpc-prototype-dev/src/hpc/hpc-irods-client/samples/Batch_Upload_Input_File_Collections.csv)

# APPENDIX B – sample data file input

[https://github.com/CBIIT/HPC\_DME\_APIs/tree/master/utils/hpc-client/samples/[Batch\_Upload\_Input\_File\_Object\_Cleversafe.csv](https://github.com/CBIIT/HPC_DME_APIs/blob/master/utils/hpc-client/samples/Batch_Upload_Input_File_Object_Cleversafe.csv)](https://ncisvn.nci.nih.gov/svn/HPC_Data_Management/branches/hpc-prototype-dev/src/hpc/hpc-irods-client/samples/Batch_Upload_Input_File_Object.csv)

[https://github.com/CBIIT/HPC\_DME\_APIs/tree/master/utils/hpc-client/samples/[Batch\_Upload\_Input\_File\_Object\_Globus.csv](https://github.com/CBIIT/HPC_DME_APIs/blob/master/utils/hpc-client/samples/Batch_Upload_Input_File_Object_Cleversafe.csv)](https://ncisvn.nci.nih.gov/svn/HPC_Data_Management/branches/hpc-prototype-dev/src/hpc/hpc-irods-client/samples/Batch_Upload_Input_File_Object.csv)

# Appendix c – sample permissions input

[https://github.com/CBIIT/HPC\_DME\_APIs/tree/master/utils/hpc-client/samples/Batch\_Permissions.csv](https://ncisvn.nci.nih.gov/svn/HPC_Data_Management/branches/hpc-prototype-dev/src/hpc/hpc-irods-client/samples/Batch_Permissions.csv)

# APPENDIX D – Executing HPC DME API with Soap UI

SoapUI is an open-source web service testing application for service-oriented architectures (SOA) and representational state transfers (REST). Its functionality covers web service inspection, invoking, development, simulation and mocking, functional testing, load and compliance testing.

## 13.1 Prerequisites

If you do not have SoapUI installed, please go to following link to download and install<https://www.soapui.org/downloads/soapui.html>

Go to: <https://ncisvn.nci.nih.gov/svn/HPC_Data_Management/branches/hpc-prototype-dev/doc> and naviage to /training/ hpc\_dme\_soapui\_tutorial. You should see following files:

* hpc-training-soapui-project.xml
* dataRegistration-async.json
* dataRegistration-sync.json
* SRR062635.filt.fastq

Once the SoapUI installation is complete, open SoapUI tool and select “Import Project” from File menu. Select “hpc-training-soapui-project.xml”.



Once the project is imported, you will see the project in the navigator. This project is created with resources to access HPC DME REST interfaces. The training project is prepopulated with multiple endpoint to access. Please select any endpoint that you have access to.



Additionally, valid NCI UserId and password are needed to authenticate with HPC DME API. Please contact HPC DME administrator (A system admin email address?) to get access on training application

## 13.2 Access HPC DME with Soap UI

To access HPC DME API, you will need to provide NCI UserId and Password through “Basic” authentication method. As shown the picture below, click on “Authentication and Security related settings” button. Select “Add New authorization” and select “Basic”. This step has to be done for every API request you make from SoapUI. Once you set it up for a request, you don’t have to recreate “Authentication and Security related settings” again.





## 13.3 Use SOAP UI to perform common use scenarios

When registering collections and data files, one must keep in mind how the data and metadata are organized in their own division or subgroup and the order data and hierarchic are managed in the logic that needs to be enforced. For instance, if the structure of “Project/Dataset/Data file” needs to be implemented, then the parent collection “Project” must be created (registered) first along with the required metadata. The subcollection “Dataset” must be created next along with that level of metadata. Only then data files under the specific dataset and project can be registered in the correct sequence.

### 13.3.1 Register a collection (Project or dataset)

“Project” is one of the collection types setup on the training endpoint. Expand “Register Collection” node, “PUT” and double click on “Register Project”. Make sure select training endpoint from the endpoint drop down. In the request section, edit “Value” column to enter desired project path. Path is a logical unique identifier to refer to a collection. For example, “/FNL\_SF\_Training/konkapv/Project1”. Click on triangle icon to submit request. You shall see the response from “Raw” tab.



Dataset is the subcollection of project (a child of project) in our sample implementation. Double click on “Register Dataset”. Make sure to select training endpoint from the endpoint drop down. In the request section, edit “Value” column to enter desired dataset path. This path should be child of an existing project. For example, “/FNL\_SF\_Training/konkapv/Project1/Dataset1”. “Click on triangle icon to submit request. You shall see the response from “Raw” tab.



### 13.3.2 Register a single data file/object into storage archive asynchronously

Prerequisite to register data object asynchronously is to have the data source location at a Globus endpoint with shared access to “ncif-hpcdm-svc” account. To create a shared access, login into [www.globus.org](http://www.globus.org) and select the endpoint where your data is located. Select the folder, and click on the icon on top right corner of the section. Click on “Share”.



Enter “Share Display Name” and click on “Create” button.



On data sharing page, select “User” radio button and click on “Search” button.



On “Search for a User” dialog, enter “ncif-hpcdm” and click on search icon. Select “Zaki Zaki” service user account.



Uncheck “Send Email” and make sure “read” permission is selected. Click on “Add permission” button.



Data file registration is done via HTTP multipart request to HPC REST interface. Tutorial package has sample metadata file (dataRegistration-async.json) to register. Edit “dataRegistration-asyn.json” package to set source fileContainerId and source fileId. “fileContainerId” is the Globus UUID and “fileId” is the full path of the file exist on Globus.

Click on “attachments” tab and click on + sign shown below to add an attachment.



Select “dataRegistration-asyn.json” you edited. Once it is attached, double click on “Content type” and enter “application/json” as the value. Double click on “ContentId” and enter “dataObjectRegistration” as the value.

Double click request value to enter unique object path value. For example, “/FNL\_SF\_Training/konkapv/Project1/Dataset1/Object1”.

Click on triangle icon to submit request. You shall see the response from “Raw” tab.

### 13.3.3 Register a single data file/object into storage archive synchronously

Data file registration is done synchronously via HTTP multipart request to HPC REST interface. Tutorial package has sample metadata file (dataRegistration-sync.json) to register. Edit “dataRegistration-syn.json” to update any metadata.

Attach metadata to the request (multipart): a) ContentType: application/json; b) ContentID: dataObjectRegistration

Click “Attachments” tab and click on the + sign shown below to add attachments mentioned above.



Select “dataRegistration-sync.json” you edited. Double click on “Content type” and enter “application/json” as the value. Double click on “ContentId” and enter “dataObjectRegistration” as the value.

Click on the + sign shown below to add attachments mentioned above. Select data object file you want to register. Double click on “ContentId” and enter “dataObject” as the value. Double click request value to enter unique object path value.

Click on triangle icon to submit request. You shall see the response from “Raw” tab.



### 13.3.4 Perform update on a metadata attribute

An existing Collection or data file metadata can be updated to add new metadata attributes or update existing metadata attribute value. We demonstrate how to perform this using the “Project” collection. Open “Update Project Metadata” resource and make sure request attribute value shows any existing project path. For example, “/FNL\_SF\_Training/konkapv/Project1”. You may edit metadata input information from media window. Click on triangle icon to submit request. You shall see the response from “Raw” tab.



### 13.3.5 Subscribe to a known event

You may subscribe to get notification emails to the events generated during data upload and download requests. This subscription is to get notification on the requests you initiate. You can also unsubscribe from some of the notifications you are already subscribed to.

Following are the valid event notifications:

DATA\_TRANSFER\_UPLOAD\_IN\_TEMPORARY\_ARCHIVE:

DATA\_TRANSFER\_UPLOAD\_ARCHIVED:

DATA\_TRANSFER\_UPLOAD\_FAILED:

DATA\_TRANSFER\_DOWNLOAD\_COMPLETED:

DATA\_TRANSFER\_DOWNLOAD\_FAILED:

Edit request media content as needed. Click on triangle icon to submit request. You shall see the response from “RAW” tab.



### 13.3.6 Generate a report

Authorized users can generate following summarized reports:

Summary report till date

Summary report by date

Summary report by DOC

Summary report by DOC and date range

ummary report by User

Summary report by User and date range

Base path report

Base path report by Date range

Collection path report

Collection path report by Date range

Edit request media content to enter valid request JSON request. Click on triangle icon to submit request. You shall see the response from “JSON” tab.



### 13.3.7 Update/assign permission

You can set permissions on the collections or data objects so that any other authenticated user can access your data. Supported permissions are OWN, READ, WRITE and NONE.

These permissions can be associated with a user or a group and are set by entity PATH. This entity can be a collection or a data file. Edit request media string to enter value entities and permissions. Click on triangle icon to submit request. You shall see the response from “JSON” tab.



### 13.3.8 Perform simple search functions

#### 13.3.8.1 Query a collection by path

Collections can be queried by its unique logical path. Double click on request path attribute value and enter a valid collection path. For example, “/FNL\_SF\_Training/konkapv/Project1”. Click on triangle icon to submit request. You shall see the response from “JSON” tab.



#### 13.3.8.2 Query a collection by metadata

Collections can be queried by their metadata. Due to the limitation of adding multiple values to a parameter within SOAP UI, you could only search by one metadata attribute. Double click on metadataQuery request attribute value field and enter search criteria.

Format is

{"a":"<attribute name>","v":"<Value>","o":"<Operator>"}

Replace <text> with the actual values. Valid values for <Operator> are

EQUAL, NOT\_EQUAL, LESS\_THAN, GREATER\_THAN, LIKE

Example:

{"a":"name","v":"%training%","o":"LIKE"}

Click on triangle icon to submit request. You shall see the response from “JSON” tab.



#### 13.3.8.3 Get a data object by path

Data objects can be queried by its unique logical path. Double click on request path attribute value and enter a valid data object path. Click on triangle icon to submit request. You shall see the response from “JSON” tab.



#### 13.3.8.4 Get a data object by metadata

Data objects can be queried by their metadata. Due to the limitation of adding multiple values to a parameter, you could only search by one metadata attribute. Double click on metadataQuery request attribute value field and enter search criteria.

Format is

{"a":"<attribute name>","v":"<Value>","o":"<Operator>"}

Replace <text> with the actual values. Valid values for <Operator> are

EQUAL, NOT\_EQUAL, LESS\_THAN, GREATER\_THAN, LIKE

Example:

{"a":"name","v":"%object%","o":"LIKE"}

Click on triangle icon to submit request. You shall see the response from “JSON” tab.



### 13.3.9 Download a data file/object to Globus share

You can download a data object that you have access to into your Globus endpoint location asynchronously. The destination Globus endpoint needs be shared with “ncif-hpcdm-svc” account to write, i.e., you need to give “write” access on shared location to the Globus application account. Double click on request path attribute value and enter valid object path.

Edit destination values as needed.

Click on triangle icon to submit request. You shall see the response from “JSON” tab.



### 13.3.10 Download a data file/object to a local directory

You can download from the archive to your local file system synchronously. Double click on request path attribute value and enter valid object path. Click on triangle icon to submit request. You shall see the response from “JSON” tab.



# APPENDIX E HPC DME CLI properties

|  |  |  |
| --- | --- | --- |
| Property name | Description | Default value |
| hpc.server.url | HPC Server API url. This is where REST interfaces are running on the server | <https://hpcdmeapi.nci.nih.gov> |
| hpc.default.globus.endpoint | Globus endpoint to transfer from |  |
| hpc.collection.service | Collection service REST resource name. You don’t need to change this value unless it is changed on the server. | collection |
| hpc.dataobject.service | Data object REST resource name. You don’t need to change this value unless it is changed on the server. | dataObject |
| hpc.error-log.dir | Path of the error log file to be written | . |
| hpc.ssl.keystore.path | Path of the keystore used for 2-way SSL connectivity with HPC Server. If the value is missing, one way SSL connection is used. | hpc-client/keystore/keystore-uat.jks |
| hpc.ssl.keystore.password | Password for the keystore. You don’t have to change this value. If the value is missing, one way SSL connection is used. | hpc-server-store-pwd |
| hpc.login.credentials | Location of the file with user credentials (user name and password). Credentials should be in format of <userId:password> |  |
| hpc.job.thread.count | Number of concurrent threads to process batch input file. Setting up large number of threads may not necessarily improve processing time. It depends on various factors like client machine processing speed, network bandwidth, server response. | 5 |
| globus.nexus.url | Globus Nexus URL. You don’t have to change this property value. | nexus.api.globusonline.org |
| globus.url | Globus URL for asynchronous data registration. You don’t have to change this property value. | [www.globusonline.org](http://www.globusonline.org) |
| hpc.login.token | HPC USER login token file location. You don’t have to change this property value. | tokens/hpcdme-auth.txt |
| hpc.globus.login.token | HPC USER Globus login token file location. You don’t have to change this property value. | tokens/globus-auth.txt |