*HPC Data MANAGEMENT*

User guide

Version *1.0*

*12/12/2016*

**Version History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version Number** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Description of Change** |
| 1.0 | Prasad Konka | 2/9/2016 |  |  | Initial Draft |
| 1.0 | Zhengwu Lu | 2/24/2016 |  |  | Comments |
| 1.0 | George Zaki | 2/25/2016 |  |  | Comments |
| 1.0 | Prasad Konka | 2/25/2016 |  |  | Updates |
| 1.0 | George Zaki | 2/25/2016 |  |  | Updates |
| 1.0 | Prasad Konka | 5/31/2016 |  |  | Comments from Zhengwu |
| 1.0 | George Zaki | 7/21/2016 |  |  | Added the curl description |
| 1.0 | Prasad Konka | 8/18/2016 |  |  | Prerequisites |
| 1.0 | Prasad Konka | 9/28/2016 |  |  | Integrated search |
| 1.0 | Zhengwu Lu | 11/7/2016 |  |  | Adding hierarchical search context |
| 1.0 | George Zaki | 11/22/2016 |  |  | Edited the hierarchical search documentation and curl commands |
| 1.0 | Zhengwu Lu | 11/23/2016 |  |  | Review and revise hierarchical search note on collections search and overall minor editing |
| 1.0 | Zhengwu Lu | 12/07/2016 |  |  | Revise hierarchical search default and expected implementation behavior |
| 1.0 | Zhengwu Lu | 12/12/2016 |  |  | Remove simple search as it is included in the general compound search implementation |
| 1.0 | Prasad Konka | 12/19/2016 |  |  | Updated server URL |
| 1.0 | George Zaki | 4/7/2017 |  |  | Added the command line utilities section |

TABLE OF CONTENTS

[1. Purpose 5](#_Toc479335440)

[2. Introduction 5](#_Toc479335441)

[2.1. What is HPC DME ? 5](#_Toc479335442)

[2.2. Intended Users 5](#_Toc479335443)

[2.3. Accessing HPC DME 5](#_Toc479335444)

[3. Pre-Requisites 6](#_Toc479335445)

[3.1 Account with HPC DME 6](#_Toc479335446)

[3.2 Globus Account for Asynchronous data transfers 6](#_Toc479335447)

[3.3 Pre-Requisites for HPC Client 6](#_Toc479335448)

[3.3.1 Java 6](#_Toc479335449)

[3.3.2 Download HPC Client 7](#_Toc479335450)

[4. HPC DME Overwiew 7](#_Toc479335451)

[4.1 Business Rules and characteristics of Collections 8](#_Toc479335452)

[4.2 Business Rules and characteristics of Data objects 9](#_Toc479335453)

[4.3 Metadata 9](#_Toc479335454)

[4.4 Data transfer 10](#_Toc479335455)

[4.5 User Authentication 11](#_Toc479335456)

[5. HPC DME API OverView 11](#_Toc479335457)

[6. HPC DME client Overwiew 12](#_Toc479335458)

[6.1 HPC DME Batch Client 12](#_Toc479335459)

[6.1.1 Set HPC batch client properties 13](#_Toc479335460)

[6.1.2 Running HPC Batch Client 14](#_Toc479335461)

[6.1.1 Error handling 25](#_Toc479335462)

[6.2 Acessing HPC DME API with CURL 25](#_Toc479335463)

[6.2.1 Setting up the environment 25](#_Toc479335464)

[6.2.2 Executing CURL commands 25](#_Toc479335465)

[6.3 Acessing HPC DME API USING CMD LINE UTILITIES 27](#_Toc479335466)

[6.3.1 One time setup 28](#_Toc479335467)

[6.3.2 Generate a token 28](#_Toc479335468)

[6.3.3 Register a collection 28](#_Toc479335469)

[6.3.4 Register a dataobject 28](#_Toc479335470)

[6.3.4.1 Synchronously (From file system) 28](#_Toc479335471)

[6.3.4.2 ASynchronously (From Globus) 29](#_Toc479335472)

[6.3.5 Get a collection metadata 29](#_Toc479335473)

[6.3.6 get a dataobject metadata 29](#_Toc479335474)

[7. APPENDIX A – sample collection input 29](#_Toc479335475)

[8. APPENDIX B – sample data file input 29](#_Toc479335476)

[9. Appendix c – sample permissions input 29](#_Toc479335477)

# Purpose

The user guide describes the functional capabilities of batch utility to register collections (i.e., datasets, projects, or folders) and data objects (i.e. files or folders) with HPC data management environment. The user guide further details the prerequisites needed for setting up the HPC data management environment (DME)**.**

# 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 huge amounts of data, produced from high performance computing systems. HPC DME provides capabilities for storing, managing, transferring and sharing huge amounts of data across different systems securely and efficiently.

Users can store data objects 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 not only to help identify the data but also enhancing 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 NCI account holders at this point of time.

## Accessing HPC DME

HPC DME can be accessed through its Service API or through its client interface. HPC DME Service APIs are developed based on REST standard. 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.

The following is the HPC DME Service API URL:

<https://hpcdmeapi.nci.nih.gov/><Resource Name>

HPC DME can also be accessed through its interactive client interface. The following are pre-requisites to access client interface.

# Pre-Requisites

## 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 of using the HPC DME. If you are a HPC DME administrator, please see HPC Server API specification for user registration details.

## Globus Account for Asynchronous data transfers

HPC DME supports transferring data from a Globus endpoint to HPC DME Archive storage. If you are working at FNLCR and you want to register data hosted at ISILON asynchronously, you would need to request for MOAB account to get access to

Please refer to the link below to submit a request:

<https://ncifrederick.cancer.gov/isp/abcc/access-request/>

However, using MOAB and Globus is not necessarily a prerequisite to deposit data files (objects) into the Cleversafe object store or any other storage medium like isilon storage system. Generally speaking, Using Globus and the GridFTP transfer is recommended for transferring large data objects (greater than or equal to 5 GB) asynchronously.

## Pre-Requisites for HPC Client

### 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.

### Download HPC Client

Download HPC client bundle from the following location and extract into a folder.

<https://ncisvn.nci.nih.gov/svn/HPC_Data_Management/branches/hpc-prototype-dev/dist/hpc-client-1.0.zip>

After extracting the bundle, you will see the following files/folders.

1. hpc.properties
2. hpc-cli-1.0.jar
3. lib (folder)
4. keystore (folder)
5. samples (folder)

# 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 large data objects with no-loss of data. 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 by default uses Globus to perform asynchronous data transfer between Globus endpoints. HPC DME pluggable architecture allows both these implementations to be replaced with alternatives easily while keeping its APIs unchanged. The basic features 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 or folders. Data object is defined as an individual file or a set of files in a folder. Collections can be of different types for example, a dataset can be linked to a sequencing project, imaging study or an analysis project etc., or a project can be an entity (umbrella project) indicating a scientific ‘study’.

## Business Rules and characteristics of Collections

* Each collection can be referred 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 a type of one of these predefined types – Project, Dataset, Folder. 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 has its own required set of metadata which needs to be submitted at the time of Project registration. This required metadata can be configured by a system 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 achieved using the logical path of a given collection.

## Business Rules and characteristics of Data objects

* A data object can be a single data file or a folder compressed or uncompressed.
* Each data object can be referred with a logical path (“object\_path”) which is unique across HPC DME.
* A parent collection should 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 has associated required metadata which is required to be submitted at the time of data object registration.
* Users have an option of adding new metadata variables to the required metadata associated with each data object.

## 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.

## Data transfer

HPC DME can register data objects from Globus endpoint source or from a file system. If the transfer source location is Globus, HPC DME requires user to share transfer data with “ncif-hpcdm-svc” service account. This can be done by going to [www.globus.org](http://www.globus.org) and select the folder to share. Globus do not support sharing individual files. If you have a single file to register, please create a folder for it and share the folder. The data transfer from Globus source location is done asynchronously.



HPC DME Server API integrates with Globus Connect API to provide seamless data transfer, tracking and reporting capabilities. Upon data object registration, HPC API initiates data transfer with Globus Connect Server asynchronously. HPC API gathers data transfer details including status asynchronously and keep the information within HPC DME database for reporting purposes. Data objects can be a single file or a folder where the whole folder and its contents including subfolders are transferred to the archive system.

Note: Globus is deprecating the use of Endpoint names. It encourages using Endpoint Id. You can retrieve endpoint Id 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”.

HPC DME can also register data objects from a local file system without Globus dependency. This operation integrates with S3 API to synchronously 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 without concern if the service account has access right to those objects to be deposited.

## User Authentication

After enrolling with HPC, the users can login into HPC DME using their user id. The HPC DME validates the user passwords through the NCI LDAP system.

# HPC DME API OverView

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 the following sections. Please see HPC DME Server API specifications for details on each of these functions.

* User
  + [Enroll user into HPC DME](#_Enroll_User) (Create)
* Create Group
  + Add or remove user(s) from a group
* Collection
  + [Register Collection with its metadata into HPC DME](#_Register_Project)
  + Find Collection by path
  + [Find Collection by metadata](#_Find_Project_by_1)
* Data Object
  + [Register data object along with metadata into HPC DME](#_Register_Dataset) (Create and Update)
  + Find Data Object by path

[Find Data Object by metadata](#_Find_Dataset_by_2)

* Permissions
  + Assign permissions on HPC DME collections or data objects
* Download
  + Download data object/file to another Globus Endpoint

The following is the HPC DME Service API URL:

<https://hpcdmeapi.nci.nih.gov/><Resource Name>

# HPC DME client Overwiew

HPC DME APIs are developed on REST standard, so any client interface can be used to access these APIs confirming REST standards.

HPC DME provides an interactive client interface to process bulk data to register collections, data objects and assign permissions.

## HPC DME Batch Client

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

1. Registering collections:

Register collections and their associated metadata

1. Registering Data objects:

Register data objects and their associated metadata

1. Update Permissions:

Update users permissions on accessing HPC DME metadata

1. Search Collections:

Search for collections based on search criteria and write results into a csv/json/txt file

1. Search Data objects:

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. 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.

### Set HPC batch client properties

Before running a HPC client, you would need to update HPC client properties file – hpc.properties.

The following are the 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.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 path>\\keystore\\keystore.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 |

### Running HPC Batch 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:

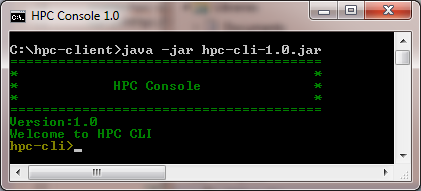
Type the following command.

java -jar hpc-cli-1.0.jar

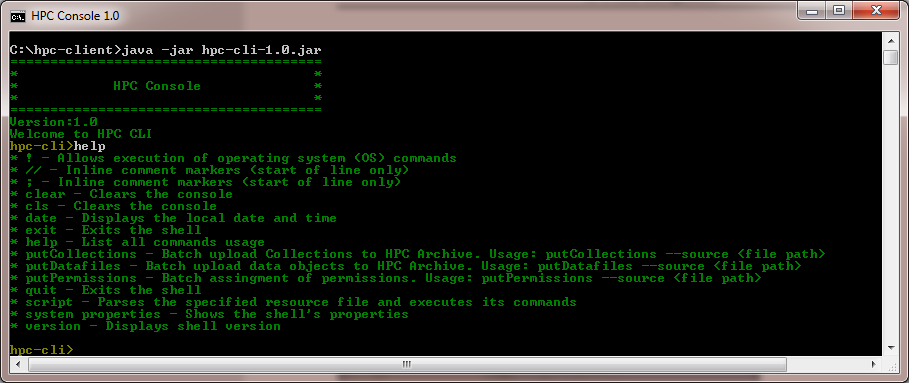
If you keep “hpc.properties” file in a different location than the location that you are running HPC client, you can use the following command.

java -Dhpc.client.properties=<path> -jar hpc-cli-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 –jar hpc-cli.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.

#### Collection Types

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 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>"

}

#### 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>

Input file for collections registration would need to provide all required metadata along with any user defined metadata. The following are the sample required metadata attributes. These attributes can be configured on HPC DME as needed by a system administrator.

The following table shows sample required attributes for a Dataset.

|  |  |  |
| --- | --- | --- |
|  | **Attribute** | **Definition** |
|  | collection\_path | Logical path of the collection. Collection path is organized in a hierarchical manner where a base parent folder represents a home folder for a division. For example, FNL\_SF\_Archive is the base folder for the sequencing facility at Frederick National Labs. So the path of the collection would be /FNL\_SF\_Archive/<collection name> |
|  | collection\_type | Collection type name (Valid values are Project, Dataset, Folder). |
|  | name | Name for the dataset of files as provided by the depositor |
|  | description | Description of dataset |
|  | source\_lab\_pi | PI of the lab of the depositor at the time of deposit |
|  | lab\_branch | Lab or Branch or Program the PI belongs to |
|  | pi\_doc | Division, Organization, Center the PI belongs to |
|  | original\_date\_created | Date the dataset was created originally |
|  | data\_creator | Person or Organization lead who created the data |
|  | phi\_content | Presence of Protected Health Information in the datasets deposited via HPC DME.  Valid values are (PHI Present, PHI Not Present, Not Specified). If no value is given, default value is “Not Specified”. |
|  | pii\_content | Presence of Personally Identifiable Information in the datasets deposited via HPC DME. Valid values are (PII Present, PII Not Present, Not Specified). If no value is given, default value is “Not Specified”. |
|  | data\_encryption\_status | If the data is encrypted or not. Valid values are (Encrypted,  Not Encrypted, Not Specified). If no value is given, default value is “Not Specified”. |
|  | data\_compression\_status | If the data is compressed or not. Valid values are (Compressed,  Not Compressed, Not Specified) If no value is given, default value is “Not Specified”. |
|  | funding\_organization | Organization Funding the generation of Data |
|  | comments | General text for internal use and reference. Optional |

The following table shows sample required attributes for a Project. Please see APPENDIX A for sample batch input file.

|  |  |  |
| --- | --- | --- |
|  | **Metadata Variable** | **Definition** |
|  | collection\_path | Logical path of the collection |
|  | collection\_type | Collection type name (Default valid values are Project, Dataset, Folder) |
|  | name | Name for the Project |
|  | description | Description of project |
|  | source\_lab\_pi | PI of the lab of the depositor at the time of deposit |
|  | lab\_branch | Lab or Branch or Program the PI belongs to |
|  | pi\_doc | Division, Organization, Center the PI belongs to |
|  | original\_date\_created | Date the dataset was created originally |
|  | project\_type | Valid values are "Unspecified", "Umbrella Project", "Sequencing", "Analysis". If not value is given, “Unspecified” value is used. |
|  | Internal\_project\_id | Internal Project Id to track |
|  | comments | General text for internal use and reference. Optional |

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 needs to be registered first before registering data files. The following is the command to register data files.

putDatafiles --source <input file path>

Input file for data file or folder registration would need to provide all required metadata along with any user defined metadata. The following are the default required metadata attributes. 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 input, HPC DME client API initiates appropriate request.

The following table shows sample required attributes for a Data file. Please see APPENDIX B for sample batch input file.

|  |  |  |
| --- | --- | --- |
|  | **Metadata Variable** | **Definition** |
|  | object\_path | Logical path of the object. Object path is organized in a hierarchical manner with the combination of parent collection path and object path. For example, /FNL\_SF\_Archive/<collection name>/<object\_name>. When the data object is registered with HPC DME and the data is transferred using Globus, logical path of the object is used to created physical path on the file system. This path could be referring to an individual file or a folder. When submitting data registration request, you could keep file/folder name same as it is at the source or change it. If the object path represents an existing folder or file, it will be overwritten by the new request. If the object path represents a folder, the data object is placed in the folder. |
|  | name | Name for the file as provided by the depositor |
|  | description | Extensible description of File |
|  | source\_lab\_pi | PI of the lab of the depositor at the time of deposit |
|  | fileId | Data Source Globus Path |
|  | fileContainerId | Data Source Globus Endpoint. Use UUID of the endpoint. |
|  | lab\_branch | Lab or Branch or Program the PI belongs to |
|  | pi\_doc | Division, Organization, Center the PI belongs to |
|  | original\_date\_created | Date the File was created originally |
|  | data\_creator | Person or Organization lead who created the data |
|  | phi\_content | Presence of Protected Health Information in the datasets deposited via HPC DME.  Valid values are (PHI Present, PHI Not Present, Not Specified). If no value is given, default value is “Not Specified”. |
|  | pii\_content | Presence of Personally Identifiable Information in the datasets deposited via HPC DME. Valid values are (PII Present, PII Not Present, Not Specified). If no value is given, default value is “Not Specified”. |
|  | data\_encryption\_status | If the data is encrypted or not. Valid values are (Encrypted,  Not Encrypted, Not Specified). If no value is given, default value is “Not Specified”. |
|  | data\_compression\_status | If the file is compressed or not. Valid values are (Compressed,  Not Compressed, Not Specified). If no value is given, default value is “Not Specified”. |
|  | funding\_organization | Organization Funding the generation of Data |
|  | comments | General text for internal use and reference. Optional |

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.

#### 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 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 resulted search records into a file.

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

**Criteria**: HPC DME metadata search API criteria is a json string with the combination of search attribute name, search attribute value and the operator to search with. The format of the criteria is as below:

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

Example:

{"a":"description","v":"%prasad%","o":"LIKE"}

Search cmd support searching on multiple criteria conditions. The following is the format:

{"a":"<attribute name>","v":"<attribute value>","o":"<Operator>"}**&&**{"a":"<attribute name>","v":"<attribute value>","o":"<Operator>"}**&&**{"a":"<attribute name>","v":"<attribute value>","o":"<Operator>"}

All the given criteria conditions are applied to the search with AND operator.

**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 for “csv” format.

**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 resulted search 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 string with the combination of search attribute name, search attribute value and the operator to search with. The format of the criteria is as below:

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

Example:

{"a":"description","v":"%prasad%","o":"LIKE"}

Search cmd support searching on multiple criteria conditions. The following is the format:

{"a":"<attribute name>","v":"<attribute value>","o":"<Operator>"}**&&**{"a":"<attribute name>","v":"<attribute value>","o":"<Operator>"}**&&**{"a":"<attribute name>","v":"<attribute value>","o":"<Operator>"}

All the given criteria conditions are applied to the search with AND operator.

**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 for “csv” format.

**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 resulted search 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

#### Perform hierarchical searches

The default hierarchical search APIs 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\_server\_API document) will search the data object 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, the users can also indicate the optional level they are basing the search on. For example, 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 there 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 at 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.

### 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.

## Acessing 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 request 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, I dump the returned response header to a file using the -D optional curl 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>

#### Registering a new user curl -H "Content-Type: application/json" -d @<user-info.json> -X PUT <server>/user -H "Accept: application/json" -D <response-header> -o <response-message.json> The input file “user-info.json” should be filled with the new user’s attributes as mentioned in the HPC\_SERVER\_API.

#### 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 HPC\_SERVER\_API, there are two methods to register data objects: asynchronously using Globus, or synchronously from the file system.

#### 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 HPC\_SERVER\_API.

#### 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 HPC\_SERVER\_API.

#### 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 should contain the attribute to be searched, and the comparison operation as mentioned in the HPC\_SERVER\_API document.

#### 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 should contain the attribute to be searched, and the comparison operation as mentioned in the HPC\_SERVER\_API document.

#### 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>

#### Download data object to a Globus endpoint curl -H "Content-Type: application/json" -d @<destination-description.json> -X POST <server>/dataObject/<object-path> /download --config $HPC\_DM\_TEST/utils/config -H "Accept: application/json" -D <response-header> -o <response-message> The “destination-description.json” file should contain the fileContainerID” and the “fileId” as described in the HPC\_SERVER\_API

#### Download data object to the file system 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”

## Acessing HPC DME API USING CMD LINE UTILITIES

Some of the curl command mentioned in the [curl section](#_Acessing_HPC_DME) 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

 To get the latest version of the utilities from git hub, execute the command:

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

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

## Generate a token

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

dm\_generate\_token

Note that token expires every two hours.

## 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 in the directory: /path/to/utils/templates. 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 dataobject

## 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 in the directory: /path/to/utils/templates.

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 should contain a valid Globus endpoint and path for your source dataObject. Also, the system account should have a read permission to the Globus endpoint as mentioned [here](#_Data_transfer). The repository contains a sample file for asynchronous dataobject-description.json as shown in this example:

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

## 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.

# APPENDIX A – sample collection input

<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://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://ncisvn.nci.nih.gov/svn/HPC_Data_Management/branches/hpc-prototype-dev/src/hpc/hpc-irods-client/samples/Batch_Permissions.csv>