*HPC Data MANAGEMENT*

API Specification

Version *1.0*

*11/07/2016*

**Version History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version Number** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Description of Change** |
| 1.0 | Prasad Konka | 1/25/2016 |  |  | DRAFT |
| 1.0 | Zhengwu Lu | 1/26/2016 |  |  | Review |
| 1.0 | Prasad Konka | 1/26/2016 |  |  | Review updates |
| 1.0 | George Zaki | 2/4/2016 |  |  | Review |
| 1.0 | Prasad Konka | 2/4/2016 |  |  | Review updates |
| 1.0 | Prasad Konka | 2/25/2016 |  |  | Review updates |
| 1.0 | George Zaki | 2/25/2016 |  |  | Review |
| 1.0 | Eran Rosenberg | 4/19/2016 |  |  | S3/Cleversafe Data Registration |
| 1.0 | Prasad Konka | 5/20/2016 |  |  | Service account updates |
| 1.0 | Prasad Konka  Zhengwu Lu | 6/2/2016 |  |  | Updates per Greg Warth comments |
| 1.0 | Prasad Konka | 7/15/2016 |  |  | System generated metadata |
| 1.0 | Prasad Konka | 8/18/2016 |  |  | Notifications, reports |
| 1.0 | Eran Rosenberg | 8/24/2016 |  |  | Get notification subscriptions |
| 1.0 | Eran Rosenberg | 10/23/2016 |  |  | Added documentation for compound metadata query search |
| 1.0 | Eran Rosenberg | 11/2/2016 |  |  | Save Query API |
| 1.0 | Zhengwu Lu | 11/7/2016 |  |  | Revise comment on complex search capability associated with the search APIs. |

TABLE OF CONTENTS

1. Purpose 5

2. Introduction 5

2.1. What is HPC DME? 5

2.2. Intended Users 5

2.3. HPC DME URL 6

3. Pre-Requisites 6

3.1. Share data with HPC DME Service account 6

4. HPC DME Overwiew 6

1.1. Data Management 6

2.1. Business Rules and characteristics of Collections 7

2.2. Business Rules and characteristics of Data objects 8

2.3. Metadata 8

2.4. System generated metadata 15

2.5. Data transfer 16

2.6. Notifications 18

2.7. Reports 19

2.8. Background processes 20

5. HPC Server API Specification 20

5.1 URL Format 20

5.2 Security 21

5.3 STATUS CODES 21

5.4 Using HPC server API 22

5.5 Enroll User 22

5.6 Update User 24

5.7 Create Group 27

5.8 Register Collection 30

5.9 Find Collection by Path 35

5.10 Find Collection by metadata 38

5.11 Find Collection by compound metadata query 40

5.12 Find Collection by compound metadata query name 43

5.13 Register Data File 44

5.14 Find data file by path 50

5.15 Find data file by metadata 52

5.16 Find data by compound metadata query 54

5.17 Find data by compound metadata query name 57

5.18 Download data file 59

5.19 Setting Entity (Collection or Data Object) Permissions 62

5.20 Subscribe to notifications 65

5.21 Get notification subscription 67

5.22 Save user query 69

5.23 Delete user query 71

5.24 Get user queries 72

5.25 Generate Report 74

6. APPENDIX A: HPC SERVER API GRAMMER 77

# Purpose

This HPC API specification is intended to help developers to understand the design and working structure of the HPC data management environment (DME). This specification will explain all the capabilities of this HPC DME and the procedures for data storage, data management, metadata collection, metadata storage and data transferring. This specification steers the users through the process of using the HPC DME for managing large data and enhancing the governance using associated metadata. The specification describes the functional capabilities of this flexibly manageable and multi-functional data archival environment and provides use cases and examples of its functionalities. The specification further details the prerequisites needed for setting up the HPC 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 large data, produced from high performance computing systems. HPC DME provides capabilities for storing, managing, transferring and sharing large data across different systems securely and efficiently.

Users can store data on HPC DME for a duration defined by data storage policy, 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 DME 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 data management environment is designed to meet the data storage and data management needs of NCI cancer community. Any user with a valid NCI user account can log into the HPC DME using NCI credentials.

Note: HPC DME currently only supports NCI account holders.

## HPC DME URL

Following is the HPC DME Server API URL:

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

# Pre-Requisites

The following pre-requisite (3.1) is required only if user role requires registering data objects using Globus within the HPC DME.

## Share data with HPC DME Service account

HPC DME uses its internal service account to do data transfer from a user’s Globus endpoint to HPC data archive target. In order to perform this action, a user would have to grant read access to HPC service account “ncif-hpcdm-svc”. Please refer to HPC\_User\_Guide on how to best perform such action.

# HPC DME Overwiew

## Data Management

1. The HPC data management environment relies on backend object storage systems (Cleversafe object store) to provide a high-reliability storage and an API to manage large data objects life cycle. The data object registration system associates a label with a given managed data file or folder and captures extensible metadata for the managed data object.

The HPC DME is accessed through its application programming interfaces (APIs) to enable users’ scientific data management activities. At a high level, there are two important components in HPC data management environment. 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 supports multiple types of data transfers such as Amazon S3 and Globus Connect. It uses Globus to perform asynchronous data transfer between Globus endpoints. It also supports synchronous data transfers from an accessible file system to Cleversafe object store. HPC DM 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 can 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. The associated metadata can be used as search criteria to identify dataset(s) through HPC DME API.

The HPC data management implementation provides users flexibility to define collection types and metadata attributes. Collections in HPC DME can be Projects, Datasets, Samples and Runs. A data object is an individual file or a set of files in a folder. Collections can be of different types. Each collection can have multiple subcollecitons if necessary. For example, a dataset can be a subcollection of a sequencing project, imaging study or an analysis project. 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. Additional collection types such as Runs, Samples may be custom added and configured in the system through metadata configuration. Please consult your HPC DME system administrator for details.
* 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 its registration. This required metadata can be configured by a system administrator through updating the metadata policy file on the server. Only system admin can update this metadata policy file on the server. There is no UI or API to update the policy file at this time. Please refer to HPC\_Admin\_Guide for more details.
* 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 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 under the collection.
* 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 which describes the actual data such as the date and origin of creation, its contents, its condition, processing it has gone through and associations to other objects etc. 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.

One of the primary functions of HPC DME is to connect unstructured data with metadata. Metadata may be attached to files, folders and collections. HPC DME stores metadata in the form of “triples” to its relational database. The triples consist of an attribute field, a value field, and a unit field. The content of each of these fields can be independently defined and applied. This Metadata can be changed and updated through the life cycle of each data object.

The following table shows default required Metadata for a dataset in the HPC DME. This list can be modified by a system administrator to add, update or remove any of these attributes. Details on updating metadata attributes configuration is detailed in the section below.

**Metadata for Dataset:**

The following table shows default required Metadata for a Dataset in the HPC DME.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Metadata Variable** | **Definition** | **Default** |
| **Administrative Metadata** | | | |
|  | collection\_type | Collection type name (Default 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 | Date the dataset was deposited |
|  | data\_creator | Person or Organization lead who created the data | Not Specified |
|  | phi\_content | Presence of Protected Health Information in the datasets deposited via HPC DME.  Valid values are (PHI Present, PHI Not Present, Not Specified) | 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) | Not Specified |
|  | data\_encryption\_status | If the data is encrypted or not. Valid values are (Encrypted,  Not Encrypted, Not Specified) | Not Specified |
|  | data\_compression\_status | If the data is compressed or not | Not Specified |
|  | funding\_organization | Organization Funding the generation of Data | Not Specified |
| **Division/Center specific metadata** | | | |
|  | comments | General text for internal use and reference |  |

The following table shows default required Metadata for a Project in the HPC DME.

**Metadata for Project:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Metadata Variable** | **Definition** | **Default** |
| **Administrative Metadata** | | | |
|  | 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 | Date the dataset was deposited |
|  | project\_type | Default valid values are "Unspecified", "Umbrella Project", "Sequencing", "Analysis" |  |
| **Division/Center specific metadata** | | | |
|  | Internal\_project\_id | Internal Project Id to track |  |
|  | comments | General text for internal use and reference |  |

**System Generated Metadata Variables for any Collection:**

|  |  |  |
| --- | --- | --- |
| Id | local identifier (serves as foreign key to connect to other metadata in data management system |  |
| uuid | universal dataset identifier (reserved for future use: default is ‘unspecified’) |  |
| create\_date | Date the project, dataset or file was registered with HPC DME. |  |
| update\_date | Date the project, dataset or file was updated with HPC DME. |  |
| registerd\_by | User depositing the dataset |  |
| registered\_by\_name | Name of the person registering the dataset |  |
| registered\_by\_doc | Division, Organization, Center of the user registering the dataset |  |

The following table shows default required Metadata for a data object in the HPC DME.

**Metadata for Data object/file:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Metadata Variable** | **Definition** | **Default** |
| **Administrative metadata** | | | |
|  | 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 | Auto Populated |
|  | 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 | Date the file was deposited |
|  | data\_creator | Person or Organization lead who created the data | Not Specified |
|  | phi\_content | Presence of Protected Health Information in the datasets deposited via HPC DME.  Valid values are (PHI Present, PHI Not Present, Not Specified) | 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) | Not Specified |
|  | data\_encryption\_status | If the data is encrypted or not. Valid values are (Encrypted,  Not Encrypted, Not Specified) | Not Specified |
|  | data\_compression\_status | If the file is compressed or not | Not Specified |
|  | funding\_organization | Organization Funding the generation of Data | Not Specified |
| **Division/Center specific metadata** | | | |
|  | comments | General text for internal use and reference |  |
| **System Generated Metadata Variables for any Data object / file (Automated)** | | | |
|  | id | local identifier (serves as foreign key to connect to other metadata in data management system |  |
|  | uuid | universal dataset identifier (reserved for future use: default is ‘unspecified’) |  |
|  | dataset\_signature\_md5 | MD5 checksum |  |
|  | Dataset\_signature\_sha1 | SHA1 checksum |  |
|  | Data\_size | Size of the dataset being deposited |  |
|  | create\_date | Date the project, dataset or file was registered with HPC DME. |  |
|  | update\_date | Date the project, dataset or file was updated with HPC DME. |  |
|  | registerd\_by | User depositing the dataset |  |
|  | registered\_by\_name | Name of the person registering the dataset |  |
|  | registered\_by\_doc | Division, Organization, Center of the user registering the dataset |  |
|  | source\_file\_container\_id | The container ID of the data source. | Globus Only |
|  | source\_file\_id | The file id of the data source. | Globus Only |
|  | archive\_file\_container\_id | The container ID of the data object in the Archive |  |
|  | archive\_file\_id | The file ID of the data object in the Archive |  |
|  | data\_transfer\_request\_id | The data transfer request ID | Globus Only |
|  | data\_transfer\_status | The status of the data transfer |  |
|  | data\_transfer\_type | The method used to transfer the data |  |
|  | Source\_file\_size | The file size of the data object source | Globus Only |

HPC DME provides a flexible way to configure metadata attributes through a policy file. Following is an example of the policy file.

|  |
| --- |
| **Sample Policy File** |
| {"HpcMetadataValidationRules": {  "collectionMetadataValidationRules": [  {  "attribute": "Collection type",  "mandatory": true,  "validValues": [  "project",  "dataset",  "folder"  ],  "ruleEnabled": true,  "DOC": "DOC-NAME"  },  {  "attribute": "Project name",  "mandatory": true,  "collectionType": "project",  "ruleEnabled": true,  "DOC": "DOC-NAME"  },  {  "attribute": "Project type",  "mandatory": true,  "defaultValue": "Unspecified",  "collectionType": "project",  "validValues": [  "Umbrella Project",  "Sequencing",  "Analysis"  ],  "ruleEnabled": true,  "DOC": "DOC-NAME"  }, |

* attribute: Metadata attribute name
* mandatory: Flag to indicate if it is required or not. Valid values are “true”, “false”
* defaultValue: Default value of the attribute if no value is given
* collectionType: Collection type name applicable for this attribute rule. Only one value is allowed.
* validValues: List of valid values for this attribute
* ruleEnabled: Flag to indicate if this rule is enabled or not. Valid values are “true” or “false”. If value is set to “false”, this rule will not be evaluated during validation process.
* DOC: Division name applicable. This rule will only be active for the given DOC. This attribute is not active now.

## System generated metadata

Apart from user provided metadata, HPC DME generates system metadata to track information for security and reporting purposes. This read only metadata can be viewed authorized users. Metadata policy file mentioned in the section above, has configuration information for system metadata. Following are the system metadata attributes supported at this time. System Administrator may remove some of these attributes from the list as needed. Removing an attribute from the policy would result not recording that attribute as part of collection or data object metadata.

**System Metadata for Collections**

|  |  |
| --- | --- |
| Attribute Name | Description |
| uuid | Collection unique identifier |
| registered\_by | Collection registered by UserId |
| registered\_by\_name | Collection registered by user name |
| registered\_by\_doc | Collection registered by user DOC |

**System Metadata for Data objects**

|  |  |
| --- | --- |
| Attribute Name | Description |
| uuid | Collection unique identifier |
| registered\_by | Collection registered by UserId |
| registered\_by\_name | Collection registered by user name |
| registered\_by\_doc | Collection registered by user DOC |
| source\_file\_container\_id | Source data object container identifier (Ex: Globus endpoint Id) |
| source\_file\_id | Source data object path |
| archive\_file\_container\_id | Archive data object container identifier |
| archive\_file\_id | Archive data object path |
| data\_transfer\_request\_id | Data transfter request identifier |
| data\_transfer\_status | Data transfer status |
| data\_transfer\_type | Data transfer type |
| source\_file\_size | Source file size |
| archive\_caller\_object\_id | archive\_caller\_object\_id is optional system metadata. It is populated when "callerObjectId" is set as part of object registration request. "callerObjectId" string is additional path information to calculate destination path. By default it is "/", so user do not have to provide it. |

## Metadata Heirarchy

Metadata can be added to a collection or a data file. These collections or data files can be logically structured in a hierarchical manner. A hierarchy is typically viewed as a pyramid and can be depicted with a tree structure. In order to search for a collection or a data file by its metadata or its parent metadata, by default, HPC DME custom solution inherits parent metadata to its children. Child entities can always override inherited parent metadata attributes. This approach facilitates advanced and efficient search capabilities where you can find data files based on its own or its parent metadata attributes. For example, you can find all data files (child) for a project (parent) with name “xyz”.

HPC DME also supports limiting your search to a particular hierarchy level. For example, if your data hierarchy is logically structures as /PI\_lab/Project/Run/Sample/Datafile:

* All PI\_lab metadata is inherited to Project and the level of this hierarchy is 5
* All Project metadata is inherited to Run and the level of this hierarchy is 4
* All Run metadata is inherited to Sample and the level of this hierarchy is 3
* All Sample metadata is inherited to Datafile and the level of this hierarchy is 2
* The level of Datafile hierarchy is 1.

When you are searching for an entity (collection or data file), you could limit your search to a particular level using comparison operators like *EQUAL, NOT\_EQUAL, NUM\_LESS\_THAN, NUM\_LESS\_OR\_EQUAL, NUM\_GREATER\_OR\_EQUAL.*

Using ‘level’ and ‘levelOperator’ as part of your search is optional. If not provided, the query will match metadata found at any level.

## Data transfer

HPC DME is configured to work with both synchronous and asynchronous data transfers. HPC DME is configured to use Cleversafe object store as its data archival storage. Data from a local or shared location can transferred to HPC data archive system synchronously. For large data object data transfer, it is recommended to host your data at a Globus endpoint so that the transfer can be asynchronous. Please see <https://www.globus.org/globus-connect-server> for more details. HPC DM Server API integrates with Globus Connect API to provide seamless data transfer, tracking and reporting capabilities. Since there is no Globus connect plugin to work with Cleversafe directly, HPC DME provides 2-hop solution to support data transfer from/to a Globus endpoint to/from HPC DME Cleversafe data store. As depicted in the diagram below, to enable 2-hop transfer, HPC DME uses temporary storage to stage data while transferring. Stage data gets cleaned as soon as the data transfer is complete.



Upon dataset registration, HPC DME initiates data transfer with Globus Connect Server asynchronously and keep the information within HPC DM database for reporting purposes. If the data transfer is done with 2-hop approach, HPC DME uses following status codes to accurately represent data transfer status.

**IN\_PROGRESS\_TO\_TEMPORARY\_ARCHIVE**: This is the status code while data is transferred from source location to HPC DME temporary storage.

**IN\_TEMPORARY\_ARCHIVE:** This is the status code if the data is transferred to HPC DME temporary storage and it is waiting to be transferred to archive storage by HPC DME.

**IN\_PROGRESS\_TO\_ARCHIVE:** This is the status code while data is transferred from HPC DME temporary storage to HPC DME archive storage.

**ARCHIVED**: This is the status code if the data is transferred to HPC DME archive storage.

**FAILED**: This is the status code if the data is transferred is failed for any known reasons.

**UNKNOWN**: This is the status code if the data is transferred is failed for any unknown reasons.

HPC DME supports synchronous data transfer from a source with no Globus endpoint. As depicted in the diagram below, HPC DME client initiate synchronous data registration request where data is transferred to the archive storage via HPC DME server.



HPC DME background process checks for any “IN\_TEMPORARY\_ARCHIVE” objects and initate data transfer to archive storage. By default, this background process runs every 60 seconds and it is configurable. Background process will update data transfer status with HPC DME and clean up data from stage area.

## Notifications

HPC DME generates different events while processing data upload, download and sharing requests. Authorized users can subscribe to these event notifications via email or text. Whenever these events happen, HPC background process pushes out notifications to subscribed users.

Following event types can be subscribed by users:

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

This event is generated when uploaded data is moved to temporary storage before moving it into archive storage. HPC DME uses 2-hop upload when using asynchronous data transfer from a Globus endpoint. As 1st hop, HPC DME transfers data from source Globus endpoint to HPC DME stage globus endpoint. Once the transfer is successful, 2nd hop of tranfering data to archive storage is initiated.

**DATA\_TRANSFER\_UPLOAD\_ARCHIVED:**

This event is generated when uploading data to the archive storage request is completed.

**DATA\_TRANSFER\_UPLOAD\_FAILED:**

This event is generated when uploading data to the archive storage request is failed.

**DATA\_TRANSFER\_DOWNLOAD\_COMPLETED:**

This event is generated when downloading data from the archive storage request is completed.

**DATA\_TRANSFER\_DOWNLOAD\_FAILED:**

This event is generated when downloading data from the archive storage request is failed.

**USAGE\_SUMMARY\_REPORT:**

This event is generated when the background process generates data archive system usage summary report.

**USAGE\_SUMMARY\_BY\_WEEKLY\_REPORT:**

This event is generated when the background process generates data archive system weekly usage summary report.

## Reports

Users of HPC DME can generate different summary reports useful to analyze its usage. Default HPC DME report provides following information:

|  |
| --- |
| Total registered users:  Total size of the data: bytes  Largest file: bytes  Average file size: bytes  Total number of objects:  Total number of collections:  Project:  Dataset:  ..  Total number of metadata attributes:  Total number of files with size:  < 1MB:  > 1MB and < 10MB:  > 10MB and < 50MB:  > 50MB and < 100MB:  > 100MB and < 500MB:  > 500MB and < 1GB:  > 1GB and < 10GB:  > 10GB: |

HPC DME provides REST API to generate following reports.

**USAGE\_SUMMARY\_REPORT:**

This report provides summarized report from inception to till date.

**USAGE\_SUMMARY\_BY\_WEEKLY\_REPORT:**

This report provides summarized report for past 1 week from today.

**USAGE\_SUMMARY\_BY\_DOC\_REPORT:**

This report provides summarized report for a given DOC(S) from inception to till date.

**USAGE\_SUMMARY\_BY\_DOC\_BY\_WEEKLY\_REPORT:**

This report provides summarized report for a given DOC(S) for past 1 week from today.

**USAGE\_SUMMARY\_BY\_USER\_REPORT:**

This report provides summarized report for a given user(s) from inception to till date.

**USAGE\_SUMMARY\_BY\_USER\_BY\_WEEKLY\_REPORT:**

This report provides summarized report for a given user(s) for past 1 week from today.

## Background processes

HPC DME Server runs background processes to synchronize some of its action with user actions. Timing of these background processes can be configured through hpc-server-\*.properties file. It is recommended not to change these settings unless there is a need. The default timings are set to avoid any race conditions with system resources. Following are the properties from hpc-server-\*.properties file.

# Runs daily at 12AM

#hpc.scheduler.cron.summaryreport.delay=0 0 24 1/1 \* ?

# Runs Weekly at 12AM

#Runs every minute at 10 secs

#hpc.scheduler.cron.processevents.delay=10 0/1 \* \* \* ?

#Runs every minute at 15 secs

#hpc.scheduler.cron.updateDataTransferUploadStatus.delay=15 0/1 \* \* \* ?

#Runs every minute at 25 secs

#hpc.scheduler.cron.processTemporaryArchive.delay=25 0/1 \* \* \* ?

#Runs every minute at 35 secs

#hpc.scheduler.cron.cleanupDataTransferDownloadFiles.delay=35 0/1 \* \* \* ?

# HPC Server API Specification

Server API can be categorized into following sections. HPC DME Server API specifications will be revised as more functionality or stabilization is added.

* User
  + [Enroll user into HPC DME](#_Enroll_User) (Create)
  + Get User
  + Update an existing User
* Collection
  + [Register Collection with its metadata into HPC DME](#_Register_Project)
  + Find Collection by path
  + [Find Collection by metadata](#_Find_Project_by_1)
* Data file
  + [Register data file 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 iRODS resources
* Download
  + Download data object/file
* Notifications
  + Subscribe to notifications
* Reports
  + Generate reports

## URL Format

HPC Server API URL is always relative to following HPC Server base URL.

http://<ip address:port>/hpc-server/

## Security

Each HPC Server API call is secured by authentication and authorization. HPC Server API is configured to support BASIC authentication credentials over HTTPS. Given credentials are verified against NCI LDAP for authentication. An authenticated User is verified with HPC security system for authorization.

Users of HPC API interface are required to enroll into HPC DME to start with. Please contact HPC DME system or group administrator to enroll you with the application.

HPC Server API internally interacts with iRODS Jargon API to work with iRODS securely. iRODS Jargon APIs are protected with PAM *(Pluggable Authentication Modules)* authentication via NCI LDAP credentials.

## STATUS CODES

HPC API returns following error object in case of any exceptions.

*{"gov.nih.nci.hpc.dto.error.HpcExceptionDTO": {*

*"errorType": "HPC Error Type",*

*"message": "Error Message",*

*"stackTrace": "Error Stacktrace”*

*}}*

Following are HPC Error Types:

SPRING\_CONFIGURATION\_ERROR,

INVALID\_REQUEST\_INPUT,

UNAUTHORIZED\_REQUEST,

REQUEST\_AUTHENTICATION\_FAILED,

REQUEST\_REJECTED,

DATABASE\_ERROR,

DATA\_TRANSFER\_ERROR,

UNEXPECTED\_ERROR;

Each of HPC Error codes are mapped following HTTP codes.

|  |  |  |
| --- | --- | --- |
| HTTP Code | Status | Description |
| 200 | OK | Query request is processed successfully. Query result may include partially successful results in case of batch processing. |
| 201 | Created | Created requested resource |
| 400 | Bad Request | Request could not be understood by the HPC server due to malformed syntax. |
| 401 | Unauthorized | User authentication is required. The request either did not provide correct credentials or the user does not have permission. |
| 500 | Internal Server Error | Internal error was encountered on the HPC server. This will be a non-recoverable error. Details are provided in message body. |

## Using HPC server API

HPC Server API follows REST specification. Any client program or language supporting REST specification can be used to interact with HPC API. Each of the API specifications given below has example Java client code via accessing HPC REST API interfaces. Client interface like SOAPUI (<http://www.soapui.org/>) can also be used to test HPC Server APIs. Installing SOAPUI instructions can be found at <http://www.soapui.org/getting-started/installing-soapui/installing-on-windows.html>

## Enroll User

|  |  |
| --- | --- |
| Title | Enroll User |
| Description | Enroll user into HPC system. In order to use HPC DME, a user must be enrolled into the DME. Only users with “SYSTEM\_ADMIN”, “GROUP\_ADMIN” roles are authorized to run this API.  “userRole” attribute is optional with the request. If not given, user is enrolled as “USER” role. Valid values are “USER”, “SYSTEM\_ADMIN” and “GROUP\_ADMIN”. There is an issue with iRODS API to create “GROUP\_ADMIN” role. Work around to update user role through iCommand interface following the registration with this API. |
| URL | /user |
| Method | PUT |
| Acceptable request representation | application/json  application/xml |
| Available response representation |  |
| URL Params | HpcUserRegistrationDTO |
| Data Params | **JSON:**  {  "nciAccount": {  "userId": "jdoe",  "firstName": "John",  "lastName": "Doe",  “DOC” : “FNLCR”  },  "userRole": “USER”  }  **XML:**  <?xml version="1.0" encoding="UTF-8" ?>  <nciAccount>  <userId>jdoe</userId>  <firstName>John</firstName>  <lastName>Doe</lastName>  </nciAccount>  <userRole>  USER  </userRole> |
| Success Response | HTTP/1.1 201 Created  Location: http://localhost:7737/hpc-server/user/u1c06009i |
| Error Response | **UserId already exists:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {"gov.nih.nci.hpc.dto.error.HpcExceptionDTO":{"errorType":"REQUEST\_REJECTED","requestRejectReason":"USER\_ALREADY\_EXISTS","message":"User already exists: nihUserId = u1c06009i","stackTrace":"gov.nih.nci.hpc.exception.HpcException: User already exists: r\n"}}  **Invalid User DOC:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid DOC: FNLCR1. Valid values: [FNLCR, CCBR]",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Invalid DOC: FNLCR1. Valid values: [FNLCR, CCBR][INVALID\_REQUEST\_INPUT] "  }  **Invalid User Role:**  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid user role: USER1. Valid values: [USER, GROUP\_ADMIN, SYSTEM\_ADMIN, NOT\_REGISTERED]",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Invalid user role: USER1. Valid values: [USER, GROUP\_ADMIN, SYSTEM\_ADMIN, NOT\_REGISTERED][INVALID\_REQUEST\_INPUT]\ "  }  **Not Authorized:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "errorType": "REQUEST\_REJECTED",  "requestRejectReason": "NOT\_AUTHORIZED",  "message": "Not authorizated to update frist name, last name, DOC, role. Please contact system administrator",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Not authorizated to update frist name, last name, DOC, role. Please contact system administrator[REQUEST\_REJECTED]"  } |

## Get User

|  |  |
| --- | --- |
| Title | Get User |
| Description | System Admin can get any user registered with HPC DME. Users with GROUP\_ADMIN and USER roles can only get their account. |
| URL | /user/{UserId} |
| Method | GET |
| Acceptable request representation | application/json  application/xml |
| Available response representation |  |
| URL Params | **UserId** |
| Data Params |  |
| Success Response | HTTP/1.1 201 Created  Location: http://localhost:7737/hpc-server/user/u1c06009i |
| Error Response | **Null NCI User ID:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {"gov.nih.nci.hpc.dto.error.HpcExceptionDTO":{"errorType":"REQUEST\_REJECTED","requestRejectReason":"INVALID\_REQUEST\_INPUT","message":"Null NCI User ID","stackTrace":"gov.nih.nci.hpc.exception.HpcException: Null NCI User ID: r\n"}}  **Unauthorized access request:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "errorType": "UNAUTHORIZED\_REQUEST",  "message": "Unauthorized access request",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Unauthorized access request "  }  **Failed to get a user:**  {  "errorType": "DATABASE\_ERROR",  "message": "Failed to get a user: ",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Failed to get a user\ "  } |

## Update User

|  |  |
| --- | --- |
| Title | Update User |
| Description | Update an existing user in the HPC system.   * Only SYSTEM\_ADMIN can update any user. * The update request reports an error if it contains information that is not authorized to be updated (e.g, dataManagementUserType "USER" is calling the request to update another user, or a self update request from a user except system\_admin). * System\_admin user type is not allowed to downgrade self role. |
| URL | /user/<userId> |
| Method | POST |
| Acceptable request representation | application/json  application/xml |
| Available response representation |  |
| URL Params | HpcUserRegistrationDTO |
| Data Params | **JSON:**  {  "firstName": "John",  "lastName": "Doe",  "DOC" : "FNLCR",  "userRole":"SYSTEM\_ADMIN",    }  **XML:**  <?xml version="1.0" encoding="UTF-8" ?>  <firstName>John</firstName>  <lastName>Doe</lastName>  <DOC> FNLCR </DOC>  < userRole > SYSTEM\_ADMIN </ userRole > |
| Success Response | HTTP/1.1 200 OK |
| Error Response | **UserId does not exists:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "errorType": "REQUEST\_REJECTED",  "requestRejectReason": "INVALID\_NCI\_ACCOUNT",  "message": "User does not exists: nciUserId = batchAdmin16",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: User does not exists: nciUserId = batchAdmin16 "  }  **Invalid User Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid add user input",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Invalid update user input "  }  **Invalid User DOC:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid DOC: FNLCR1. Valid values: [FNLCR, CCBR]",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Invalid DOC: FNLCR1. Valid values: [FNLCR, CCBR][INVALID\_REQUEST\_INPUT] "  }  **Invalid User Role:**  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid user role: USER1. Valid values: [USER, GROUP\_ADMIN, SYSTEM\_ADMIN, NOT\_REGISTERED]",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Invalid user role: USER1. Valid values: [USER, GROUP\_ADMIN, SYSTEM\_ADMIN, NOT\_REGISTERED][INVALID\_REQUEST\_INPUT]\ "  }  **Not Authorized:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "errorType": "REQUEST\_REJECTED",  "requestRejectReason": "NOT\_AUTHORIZED",  "message": "Not authorizated to update frist name, last name, DOC, role. Please contact system administrator",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Not authorizated to update frist name, last name, DOC, role. Please contact system administrator[REQUEST\_REJECTED]"  }  **Not Authorized:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "errorType": "REQUEST\_REJECTED",  "requestRejectReason": "NOT\_AUTHORIZED",  "message": "Not authorizated to downgrade self. Please contact system administrator",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Not authorizated to downgrade self. Please contact system administrator[REQUEST\_REJECTED]"  } |

## Create/Update Group

|  |  |
| --- | --- |
| Title | Create/Update group to add or remove users to it |
| Description | Create a group and add users to it. The same API can be used to create or update an existing group to add user(s) or remove user(s). SYSTEM\_ADMIN, GROUP\_ADMIN roles are allowed to perform this action.  Following is an example request:  {  "group": "testgroup3",  "addUserIds": ["konkapv","luz6"],  "deleteUserIds": ["konkapv2"]  }  The following is an example response. Response for each user is captured and presented.  {  "group": "testgroup3",  "result": true,  "message": "Group is updated",  "userGroupResponses": [  {  "userId": "konkapv",  "result": true,  "message": "UserId: konkapv is added to group: testgroup3"  },  {  "userId": "luz6",  "result": true,  "message": "UserId: luz6 is added to group: testgroup3"  },  {  "userId": " konkapv2",  "result": true,  "message": "UserId: konkapv2 is removed from the group: testgroup3"  }  ]  } |
| URL | /group |
| Method | POST |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params | HpcGroupRequestDTO |
| Data Params | **JSON:**  {  "group": "testgroup1",  "addUserIds": ["konkapv","luz6"],  "deleteUserIds": ["konkapv2"]  }  **XML:**  <?xml version="1.0" encoding="UTF-8" ?>  <group>testgroup1</group>  <addUserIds type="array">  <value>konkapv</value>  <value>luz6</value>  </addUserIds>  <deleteUserIds type="array">  <value>konkapv2</value>  </deleteUserIds> |
|  | HTTP/1.1 200 OK |
| Error Response | **Invalid Request:**  Following error is thrown if Group name is empty of missing  HTTP/1.1 500 Bad Request  Content-Type: application/json  {  "errorType": "DATA\_MANAGEMENT\_ERROR",  "message": "Failed add iRODS user group: null or missing userGroupName",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Failed add iRODS user group: null or missing userGroupName[DATA\_MANAGEMENT\_ERROR] "  }  **Invalid Request:**  Following error is thrown if Group element is missing from the request  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Null Group request",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Null Group request”  }  **Not Authorized:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "errorType": "REQUEST\_AUTHENTICATION\_FAILED",  "message": "Access Denied: Unauthorized",  "stackTrace": "org.apache.cxf.interceptor.security.AccessDeniedException: Unauthorized "  } |

## Register Collection

Collection registration is a required process. Collections support logical hierarchical structure where a collection can have a parent and many children. Each of these colletions can represent a Project, Dataset or a folder, or any customized category of collection type. More collection types and their metadata are configurable. Based on metadata configuration, a user is mandated to provide required metadata while registering a collection.

|  |  |
| --- | --- |
| Title | Register Collection |
| Description | A collection registered with HPC DME |
| URL | /hpc-server/collection/{path} |
| Method | POST |
| Acceptable request representation | application/json  application/xml |
| Available response representation |  |
| URL Params | {path} – Logical path of the collection to identify with. |
| Data Params | **JSON:**  {  "metadataEntries": [  {  "attribute": "name",  "value": "Demo Project Name10"  },  {  "attribute": "collection\_type",  "value": "Project"  },  {  "attribute": "description",  "value": "Project desc"  },  {  "attribute": "internal\_project\_id",  "value": "ipi"  },  {  "attribute": "source\_lab\_pi",  "value": "Source Lab PI"  },  {  "attribute": "lab\_branch",  "value": "Lab / Branch Name"  },  {  "attribute": "pi\_doc",  "value": "FNLCR"  },  {  "attribute": "original\_date\_created",  "value": "10/25/2006"  }    ]  }  **XML:**  <?xml version="1.0" encoding="UTF-8" ?>  <metadataEntries>  <element>  <attribute>name</attribute>  <value>Demo Project Name10</value>  </element>  <element>  <attribute>collection\_type</attribute>  <value>Project</value>  </element>  <element>  <attribute>description</attribute>  <value>Project desc</value>  </element>  <element>  <attribute>internal\_project\_id</attribute>  <value>ipi</value>  </element>  <element>  <attribute>source\_lab\_pi</attribute>  <value>Source Lab PI</value>  </element>  <element>  <attribute>lab\_branch</attribute>  <value>Lab / Branch Name</value>  </element>  <element>  <attribute>pi\_doc</attribute>  <value>FNLCR</value>  </element>  <element>  <attribute>original\_date\_created</attribute>  <value>10/25/2006</value>  </element>  </metadataEntries> |
| Success Response | HTTP/1.1 201 Created  HPC-API-Version: 1.0.0  Location:http://fr-s-hpcdm-gp-d.ncifcrf.gov:7737/hpc-server/collection/DemoProjectName  Following is the successful response on updating an existing Collection  HTTP/1.1 200 OK  HPC-API-Version: 1.0.0  Location:http://fr-s-hpcdm-gp-d.ncifcrf.gov:7737/hpc-server/collection/DemoProjectName |
| Error Response | **Invalid Request Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Missing mandataory metadata: name",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Missing mandataory metadata: Project name[INVALID\_REQUEST\_INPUT]… "  }  **Invalid Request Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid Metadata Value: collection\_type = Project1. Valid values: [Project, Dataset, Folder]",  "stackTrace": "…"  }  **Invalid Request Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": " Path already exists as a file:",  "stackTrace": "…"  }  **Invalid Request Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": " System generated metadata can't be set/changed",  "stackTrace": "…"  }  **Invalid Request Input:**  This error is thrown if path is null or metadata is null or empty  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Null path or Invalid metadata entry",  "stackTrace": "…"  }  **Invalid Request Input:**  This error is thrown if metadata entry value is missing  HTTP/1.1 500 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "DATA\_MANAGEMENT\_ERROR",  "message": " Failed to add metadata to a collection: value is null or empty",  "stackTrace": "…"  }  **Collection Path is missing:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Failed to create directory: /",  "stackTrace": "…"  }  **Failed to create Collection Path:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "DATA\_MANAGEMENT\_ERROR ",  "message": "Failed to create a collection directory: /",  "stackTrace": "…"  }  **Failed to add metadata to collection:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "DATA\_MANAGEMENT\_ERROR ",  "message": " Failed to add metadata to a collection: /",  "stackTrace": "…"  }  **Authentication Failure:**  HTTP/1.1 401 Unauthorized  Content-Type: application/json  JSON:  {  "errorType": "REQUEST\_AUTHENTICATION\_FAILED",  "message": "Access Denied: LDAP authentication failed",  "stackTrace": "…"  } |

## Find Collection by Path

|  |  |
| --- | --- |
| Title | Find collection by Path |
| Description | Find collection by its logical path |
| URL | /hpc-server/collection |
| Method | GET |
| Acceptable request representation |  |
| Available response representation | application/json  application/xml |
| URL Params | {path} – Logical path of the collection |
| Data Params |  |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0  {"collections": {  "collection": {  "collectionId": 11580,  "collectionName": "/FNL\_SF\_Archive/Project300",  "absolutePath": "/FNL\_SF\_Archive/Project300",  "collectionParentName": "/FNL\_SF\_Archive/",  "collectionOwnerName": "konkapv",  "collectionOwnerZone": "tempZone",  "collectionMapId": 0,  "collectionInheritance": "",  "comments": "",  "info1": "",  "info2": "",  "createdAt": "2016-02-23",  "modifiedAt": "2016-02-23",  "specColType": "NORMAL"  },  "metadataEntries": [  {  "attribute": "collection\_type",  "value": "Project"  },  {  "attribute": "description",  "value": "Project desc"  },  {  "attribute": "internal\_project\_id",  "value": "ipi"  },  {  "attribute": "lab\_branch",  "value": "Lab / Branch Name"  },  {  "attribute": "name",  "value": "Demo Project Name10"  },  {  "attribute": "original\_date\_created",  "value": "10/25/2006"  },  {  "attribute": "pi\_doc",  "value": "FNLCR"  },  {  "attribute": "project\_type",  "value": "Unspecified"  },  {  "attribute": "source\_lab\_pi",  "value": "Source Lab PI"  }  ]  }} |
| Error Response | **No matching results:**  HTTP/1.1 204 No Content  Content-Length: 0  **Invalid Request Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": " Null collection path",  "stackTrace": "…"  }  **Data Management Error:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": " DATA\_MANAGEMENT\_ERROR",  "message": " Failed to get Collection: ",  "stackTrace": "…"  }  **No matching results:**  HTTP/1.1 204 No Content  Content-Length: 0  HPC-API-Version: 1.0.0 |

## Find Collection by Simple MetadATA Query

|  |  |
| --- | --- |
| Title | Find collection by metadata |
| Description | Find collection by metadata attribute name and its value. Following operators are supported to query by attribute value:  EQUAL,NOT\_EQUAL,LESS\_THAN,GREATER\_THAN,LESS\_OR\_EQUAL,GREATER\_OR\_EQUAL,NUM\_LESS\_THAN,NUM\_LESS\_OR\_EQUAL,NUM\_GREATER\_OR\_EQUAL,IN,NOT\_IN,BETWEEN,NOT\_BETWEEN,LIKE,NOT\_LIKE,SOUNDS\_LIKE  Find by metadata supports using multiple metadata attributes as search criteria. Using more than one metadata attribute results AND operation between those attributes. Our custom-developed APIs supports complex hiearchical metadata based search functions; and, metadata inheritance is also supported at this time. Development of web UI supporting searching functions are being in performed honoring inheritance of collection hierarchy. Readers are recommended to consult HPC Users Guide to find details how these can be performed. |
| URL | /hpc-server/collection |
| Method | GET |
| Acceptable request representation |  |
| Available response representation | application/json  application/xml |
| URL Params | metadataQuery={"a":"<attribute name1>","v":"<attribute value1>","o":"<comparator>"}& metadataQuery={"a":"<attribute name2>","v":"<attribute value2>","o":"<comparator>"}& metadataQuery={"a":"<attribute name3>","v":"<attribute value3>","o":"<comparator>"} |
| Data Params | metadataQuery={"a":"<attribute name1>","v":"<attribute value1>","o":"<comparator>"}& metadataQuery={"a":"<attribute name2>","v":"<attribute value2>","o":"<comparator>"}& metadataQuery={"a":"<attribute name3>","v":"<attribute value3>","o":"<comparator>"} |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0  {"collections": {  "collection": {  ..  },  "metadataEntries": [  ..  ]  }} |
| Error Response | **Data Management Error:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": " DATA\_MANAGEMENT\_ERROR",  "message": " Failed to get Collections: ",  "stackTrace": "…"  }  **Invalid Query param:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType":"INVALID\_REQUEST\_INPUT",  "message":"Invalid query operator: AEQUAL. Valid values: [ EQUAL | NOT\_EQUAL | LESS\_THAN | GREATER\_THAN | LESS\_OR\_EQUAL | GREATER\_OR\_EQUAL | NUM\_LESS\_THAN | NUM\_LESS\_OR\_EQUAL | NUM\_GREATER\_OR\_EQUAL | IN | NOT\_IN | BETWEEN | NOT\_BETWEEN | LIKE | NOT\_LIKE | SOUNDS\_LIKE ]","  stackTrace":"..”  }  **Invalid query attributes:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {"errorType":"INVALID\_REQUEST\_INPUT","message":"Invalid or empty metadata queries","stackTrace":"…"}  **No matching results:**  HTTP/1.1 204 No Content  Content-Length: 0  HPC-API-Version: 1.0.0 |

## Find Collection by compound metadata query

|  |  |
| --- | --- |
| Title | Find collection by a compound metadata query |
| Description | Find collection(s) that match a compound metadata query.  A compound query is comprised of 3 components:   * An optional list of simple metadata queries. Each simple metadata query provides a single ‘query’ to match a metadata entry: {                     "attribute": "<*metadata attribute name to query*>",                    "value": "<*metadata value to query*>”,                    "operator": “<*metadata query operator – EQUAL, NOT\_EQUAL, NUM\_LESS\_THAN,NUM\_LESS\_OR\_EQUAL,NUM\_GREATER\_OR\_EQUAL,LIKE*>”  “level” : *“<Metadata level filter. e.g /Coll\_A/Coll\_B/Coll\_C . The hierarchical metadata for ‘Coll\_C’ include all the metadata associated with ‘Coll\_C’ at level 1, ‘Coll\_B’ at level 2, and ‘Coll\_A’ at level 3>”* ,  “levelOperator” : “>Level filter operator - *EQUAL, NOT\_EQUAL, NUM\_LESS\_THAN,NUM\_LESS\_OR\_EQUAL,NUM\_GREATER\_OR\_EQUAL*  }  note: ‘level’ and ‘levelOperator’ are optional. If not provided, the query will match metadata found at any level.   * An optional list of compound query. A compound query can include a nested list of compound queries up to 10 nesting levels. * Compound query operator. Valid values are ‘ANY’ or ‘ALL’. ‘ANY’ (equivalent to logical OR) will return collections that match at least one simple query or compound query listed in the 2 lists described above. ‘ALL’ (equivalent to logical AND) will return collections that match all simple queries and compound queries in the lists mentioned above. |
| URL | /hpc-server/collection/query/compound |
| Method | POST |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params |  |
| Data Params | {     "query":    {        "operator": "ALL",        "queries":       [                    {              "attribute": "ATTR 1",              "value": "VAL 1",              "operator": "EQUAL",  “level”: 1,  “levelOperator” : “NOT\_EQUAL”           },                    {              "attribute": "ATTR 2",              "value": "VAL 2",              "operator": "LIKE"           }        ],        "compoundQueries":       [                    {              "operator": "ALL",              "queries":             [                                {                    "attribute": "ATTR 1",                    "value": "VAL 1",                    "operator": "EQUAL"                 },                                {                    "attribute": "ATTR 2",                    "value": "VAL 2",                    "operator": "LIKE"                 }              ]           },           {              "operator": "ANY",              "queries":  [              {                    "attribute": "ATTR 1",                    "value": "VAL 1",                    "operator": "EQUAL"                 },                                {                    "attribute": "ATTR 2",                    "value": "VAL 2",                    "operator": "LIKE"                 }              ]           }        ]     },     "detailedResponse": false  } |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0  {"collectionPaths": [     "/Coll\_A/Coll\_B/Coll\_C”  ]  } |
| Error Response | **Data Management Error:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": " DATABASE\_ERROR",  "message": " Failed to get Collections: ",  "stackTrace": "…"  }  **Invalid query attributes:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {"errorType":"INVALID\_REQUEST\_INPUT","message":"Invalid or empty metadata queries","stackTrace":"…"}  **No matching results:**  HTTP/1.1 204 No Content  Content-Length: 0  HPC-API-Version: 1.0.0 |

## Find Collection by compound metadata query name

|  |  |
| --- | --- |
| Title | Find collection by a compound metadata query name |
| Description | Perform search using a saved query |
| URL | /hpc-server/collection/query/compound{queryName} |
| Method | GET |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params | queryName |
| Data Params |  |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0  {"collectionPaths": [     "/Coll\_A/Coll\_B/Coll\_C”  ]  } |
| Error Response | **Data Management Error:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": " DATABASE\_ERROR",  "message": " Failed to get Collections: ",  "stackTrace": "…"  }  **Invalid query attributes:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {"errorType":"INVALID\_REQUEST\_INPUT","message":"Invalid or empty metadata queries","stackTrace":"…"}  **No matching results:**  HTTP/1.1 204 No Content  Content-Length: 0  HPC-API-Version: 1.0.0 |

## Register Data File

|  |  |
| --- | --- |
| Title | Register data file |
| Description | Register data file along with its metadata into DME. Registering a data file requires a registrar and PI to have a valid account with the DME. Registering a data file would transfer dataset from its origin endpoint or file system to a target archive storage.  Registering data file supports two data transfer types:   1. Synchronous data transfer from local file system to HPC Data Archive storage. 2. Asynchronous data transfer from a Globus endpoint to HPC Data Archive storage. HPC DME uses service accout to transfer data from your source location. You need to share data with “ncif-hpcdm-svc” before submitting your request.   Based on the given request input, the data transfer type is determined by the API.  Data file registration is done via HTTP multipart request to HPC REST interface. Following attachments are expected:   1. Attach metadata to the request (multipart)    1. ContentType: application/json    2. ContentID: dataObjectRegistration 2. Attach the file to the request (multipart) (Not needed for Globus source)    1. ContentType: application/octet-stream   ContentID: dataObject    If your data file source is a Globus endpoint location, “dataObject” attachment should not be submitted as part of the request.  If your data file source is a local file system, “source” element should be there in “dataObjectRegistration” JSON/XML submitted as part of the request. |
| URL | /hpc-server/dataObject/{urlPath} |
| Method | PUT |
| Acceptable request representation | application/json  application/xml |
| Available response representation |  |
| URL Params | {urlPath} – Logical path of the data object. Typical structure of the dataobject can be <group folder>/<project name>/<dataset name>/<dataobject name> |
| Media Type | multipart/form-data |
| Data Params | **JSON:**  Example metadata:  {  "source": {  "fileContainerId": "e1c6b3bd-6d04-11e5-ba46-22000b92c6ec",  "fileId": "/GridFTP/GridFTP\_t3/konkapv/Data1/test2.fastq"  },  "metadataEntries": [  {  "attribute": "name",  "value": "Set100"  }  ]  }  **XML:**  <?xml version="1.0" encoding="UTF-8" ?>  <source>  <endpoint>e1c6b3bd-6d04-11e5-ba46-22000b92c6ec</endpoint>  <path>/GridFTP/GridFTP\_t3/konkapv/Set100</path>  </source>  <filePath>/</filePath>  <metadataEntries>  <element>  <attribute>name</attribute>  <value>Set100</value>  </element>  </metadataEntries> |
| Success Response | HTTP/1.1 201 Created  Content-Length: 0  HPC-API-Version: 1.0.0  Location: http://fr-s-hpcdm-gp-d.ncifcrf.gov:7737/hpc-server/dataObject/tempZone%2Fhome%2FDemoProjectName3%2Fdata1 |
| Error Response | **Data file path already exists:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType":"REQUEST\_REJECTED","  requestRejectReason":"DATA\_OBJECT\_PATH\_ALREADY\_EXISTS","message":"Path already exists: /tempZone/home/DemoProjectName3",  "stackTrace":"…"}  **Both data transfer source and data attachment provided:**  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Both data transfer source and data attachment provided",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Both data transfer source and data attachment provided "  }  **No data transfer source or data attachment provided:**  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "No data transfer source or data attachment provided",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: No data transfer source or data attachment provided "  }  **Data file Path is missing:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": " Null path or dataObjectRegistrationDTO: /",  "stackTrace": "…"  }  **Invalid Data object Path:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "REQUEST\_REJECTED",  "requestRejectReason": "INVALID\_DATA\_OBJECT\_PATH",  "message": "Invalid data object path. Directory doesn't exist: /FNL\_SF\_Archive/Project-test3/D/test2.fastq",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Invalid data object path. Directory doesn't exist:..”  }  **Invalid Request Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Missing mandataory metadata: Project name",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Missing mandataory metadata: Project name[INVALID\_REQUEST\_INPUT]… "  }  **Invalid Request Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid Metadata Value: Collection type = Project1. Valid values: [Project, Dataset, Folder]",  "stackTrace": "…"  }  **Invalid Request Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Null path or Invalid metadata entry",  "stackTrace": "…"  }  **Invalid Request Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid Data Transfer Input",  "stackTrace": "…"  }  **Invalid Request Input:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": " Path already exists as a directory",  "stackTrace": "…"  }  **Data Management Error:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "DATA\_MANAGEMENT\_ERROR",  "message": " Failed to create a file",  "stackTrace": "…"  }  **Data Transfer Error:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "DATA\_TRANSFER\_ERROR",  "message": "Failed to activate endpoint: nihnci#NIH-NCI-TRANSFER",  "stackTrace": "…"  }  **Authentication Failure:**  HTTP/1.1 401 Unauthorized  Content-Type: application/json  JSON:  {  "errorType": "REQUEST\_AUTHENTICATION\_FAILED",  "message": "Access Denied: LDAP authentication failed",  "stackTrace": "…"  } |
|  |  |

## Find data file by path

|  |  |
| --- | --- |
| Title | Find data file by its logical path |
| Description | Find data file by its logical path |
| URL | /hpc-server/dataObject |
| Method | GET |
| Acceptable request representation |  |
| Available response representation | application/json  application/xml |
| URL Params | {path} – Logical path of the data file |
| Data Params |  |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0  {"dataObjects": {  "dataObject": {  "id": 10960,  "collectionId": 10950,  "collectionName": "/FNL\_SF\_Archive/Project203/Dataset203",  "absolutePath": "/FNL\_SF\_Archive/Project203/Dataset203/01-bdee-d7213b0d47e1",  "dataReplicationNumber": 0,  "dataVersion": 0,  "dataSize": 0,  "dataTypeName": "generic",  "resourceGroupName": "",  "resourceName": "demoResc",  "dataPath": "/var/lib/irods/iRODS/Vault/home/FNL\_SF\_Archive/Project203/Dataset203/01-bdee-d7213b0d47e1",  "dataOwnerName": "konkapv",  "dataOwnerZone": "tempZone",  "replicationStatus": 1,  "dataStatus": "",  "checksum": "",  "expiry": "00000000000",  "dataMapId": 0,  "comments": "",  "createdAt": "2016-02-05",  "updatedAt": "2016-02-05",  "specColType": "NORMAL"  },  "metadataEntries": [  ..  ]  }} |
| Error Response | **No matching results:**  HTTP/1.1 204 No Content  Content-Length: 0  Date: Wed, 26 Aug 2015 18:30:07 GMT  Server: Jetty(8.1.15.v20140411)  **Null or Invalid Path:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Null data object path ",  "stackTrace": "…"  } |

## Find data file by metadata

|  |  |
| --- | --- |
| Title | Find data file by metadata |
| Description | Find data file by metadata attribute name and its value. Following operators are supported to query by attribute value:  EQUAL,NOT\_EQUAL,LESS\_THAN,GREATER\_THAN,LESS\_OR\_EQUAL,GREATER\_OR\_EQUAL,NUM\_LESS\_THAN,NUM\_LESS\_OR\_EQUAL,NUM\_GREATER\_OR\_EQUAL,IN,NOT\_IN,BETWEEN,NOT\_BETWEEN,LIKE,NOT\_LIKE,SOUNDS\_LIKE  Find by metadata supports using multiple metadata attributes as search criteria. Using more than one metadata attribute results AND operation between those attributes.  Current metadata search only support searching metadata associated with each individual collection. Metadata inheritance is not supported at this time. Next version of our API will support searching collections honoring inheritance of collection hierarchy. |
| URL | /hpc-server/dataObject |
| Method | GET |
| Acceptable request representation |  |
| Available response representation | application/json  application/xml |
| URL Params | metadataQuery={"a":"<attribute name1>","v":"<attribute value1>","o":"<comparator>"}& metadataQuery={"a":"<attribute name2>","v":"<attribute value2>","o":"<comparator>"}& metadataQuery={"a":"<attribute name3>","v":"<attribute value3>","o":"<comparator>"} |
| Data Params | metadataQuery={"a":"<attribute name1>","v":"<attribute value1>","o":"<comparator>"}& metadataQuery={"a":"<attribute name2>","v":"<attribute value2>","o":"<comparator>"}& metadataQuery={"a":"<attribute name3>","v":"<attribute value3>","o":"<comparator>"} |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0  {"dataObjects": {  "dataObject": {  ..  },  "metadataEntries": [  ..  ]  }} |
| Error Response | **No matching results:**  HTTP/1.1 204 No Content  Content-Length: 0  **Invalid Query param:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {"errorType":"INVALID\_REQUEST\_INPUT","message":"Invalid query operator: AEQUAL. Valid values: [ EQUAL | NOT\_EQUAL | LESS\_THAN | GREATER\_THAN | LESS\_OR\_EQUAL | GREATER\_OR\_EQUAL | NUM\_LESS\_THAN | NUM\_LESS\_OR\_EQUAL | NUM\_GREATER\_OR\_EQUAL | IN | NOT\_IN | BETWEEN | NOT\_BETWEEN | LIKE | NOT\_LIKE | SOUNDS\_LIKE ]","stackTrace":"..”}  **Invalid query attributes:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {"errorType":"INVALID\_REQUEST\_INPUT","message":"Invalid or empty metadata queries","stackTrace":"…"}  **Failed to get data objects:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {"errorType":" DATA\_MANAGEMENT\_ERROR","message":" Failed to get data objects:","stackTrace":"…"} |

## Find data by compound metadata query

|  |  |
| --- | --- |
| Title | Find data by a compound metadata query |
| Description | Find data object(s) that match a compound metadata query.  A compound query is comprised of 3 components:   * An optional list of simple metadata queries. Each simple metadata query provides a single ‘query’ to match a metadata entry: {                     "attribute": "<*metadata attribute name to query*>",                    "value": "<*metadata value to query*>”,                    "operator": “<*metadata query operator – EQUAL, NOT\_EQUAL, NUM\_LESS\_THAN,NUM\_LESS\_OR\_EQUAL,NUM\_GREATER\_OR\_EQUAL,LIKE*>”  “level” : *“<Metadata level filter. e.g /Coll\_A/Coll\_B/Data\_A . The hierarchical metadata for ‘Data\_A include all the metadata associated with ‘Data\_A at level 1, ‘Coll\_B’ at level 2, and ‘Coll\_A’ at level 3>”* ,  “levelOperator” : “>Level filter operator - *EQUAL, NOT\_EQUAL, NUM\_LESS\_THAN,NUM\_LESS\_OR\_EQUAL,NUM\_GREATER\_OR\_EQUAL*  }  note: ‘level’ and ‘levelOperator’ are optional. If not provided, the query will match metadata found at any level.   * An optional list of compound query. A compound query can include a nested list of compound queries up to 10 nesting levels. * Compound query operator. Valid values are ‘ANY’ or ‘ALL’. ‘ANY’ (equivalent to logical OR) will return collections that match at least one simple query or compound query listed in the 2 lists described above. ‘ALL’ (equivalent to logical AND) will return collections that match all simple queries and compound queries in the lists mentioned above. |
| URL | /hpc-server/dataObject/query/compound |
| Method | POST |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params |  |
| Data Params | {     "query":    {        "operator": "ALL",        "queries":       [                    {              "attribute": "ATTR 1",              "value": "VAL 1",              "operator": "EQUAL",  “level”: 1,  “levelOperator” : “NOT\_EQUAL”           },                    {              "attribute": "ATTR 2",              "value": "VAL 2",              "operator": "LIKE"           }        ],        "compoundQueries":       [                    {              "operator": "ALL",              "queries":             [                                {                    "attribute": "ATTR 1",                    "value": "VAL 1",                    "operator": "EQUAL"                 },                                {                    "attribute": "ATTR 2",                    "value": "VAL 2",                    "operator": "LIKE"                 }              ]           },           {              "operator": "ANY",              "queries":  [              {                    "attribute": "ATTR 1",                    "value": "VAL 1",                    "operator": "EQUAL"                 },                                {                    "attribute": "ATTR 2",                    "value": "VAL 2",                    "operator": "LIKE"                 }              ]           }        ]     },     "detailedResponse": false  } |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0  {"collectionPaths": [     "/Coll\_A/Coll\_B/Data\_A”  ]  } |
| Error Response | **Data Management Error:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": " DATABASE\_ERROR",  "message": " Failed to get Collections: ",  "stackTrace": "…"  }  **Invalid query attributes:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {"errorType":"INVALID\_REQUEST\_INPUT","message":"Invalid or empty metadata queries","stackTrace":"…"}  **No matching results:**  HTTP/1.1 204 No Content  Content-Length: 0  HPC-API-Version: 1.0.0 |

## Find data by compound metadata query name

|  |  |
| --- | --- |
| Title | Find data by a compound metadata query name |
| Description | Invoke a saved query . |
| URL | /hpc-server/dataObject/query/compound/{queryName} |
| Method | GET |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params | queryName |
| Data Params |  |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0  {"collectionPaths": [     "/Coll\_A/Coll\_B/Data\_A”  ]  } |
| Error Response | **Data Management Error:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  JSON:  {  "errorType": " DATABASE\_ERROR",  "message": " Failed to get Collections: ",  "stackTrace": "…"  }  **Invalid query attributes:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {"errorType":"INVALID\_REQUEST\_INPUT","message":"Invalid or empty metadata queries","stackTrace":"…"}  **No matching results:**  HTTP/1.1 204 No Content  Content-Length: 0  HPC-API-Version: 1.0.0 |

## Download data file

|  |  |
| --- | --- |
| Title | Download/transfer data file from Archive storage to another location |
| Description | This API transfers a data file from archive storage to another accessible location, which may be a Globus endpoint to the requestor or their own accessible file system. Data object path represents the logical path of the data object registered with HPC DME. This logical path could be referring to an individual file or a folder. Destination path in the request represents the physical path on the Globus endpoint. When a user requests a download, the API lets user keep file/folder name same as it is at the source or change it. If you are downloading a file and if the destination path is a folder, the file will be downloaded into the given folder. If the destination path is a file, given destination file name will be used to save the file. |
| URL | /hpc-server/dataObject/{path:.\*}/download |
| Method | POST |
| Acceptable request representation | application/json  application/xml |
| Available response representation |  |
| URL Params | path |
| Data Params | **To request a download to a Globus endpoint**  **JSON:**  {  "destination": {  "fileContainerId": "nihnci#NIH-NCI-TRANSFER1",  "fileId": "/GridFTP/GridFTP\_t3/konkapv/Set100"  }  }  **XML:**  <?xml version="1.0" encoding="UTF-8" ?>  < destination>  <fileContainerId>nihnci#NIH-NCI-TRANSFER1</endpoint>  <fileId>/GridFTP/GridFTP\_t3/konkapv/Set100</path>  </ destination>  **To request a direct data download from Cleversafe**  **JSON:**  **{**  **}**  **XML**  <?xml version="1.0" encoding="UTF-8" ?> |
| Success Response | **Globus download response**  HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0  {     "requestId": "c9930828-0a35-11e6-a73e-22000bf2d559",     "destination":    {        "fileContainerId": "eranrosenberg#hpc-test",        "fileId": "/~/Development/Tools/globus/drop/eran-data-object-file"     }  }  **Cleversafe data download response**  <data contentType="application/octet-stream" contentLength="562728">  -- Binary Data – |
| Error Response | **Invalid Path:**  This error is thrown if an invalid data object path is given  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Data object not found: <Path>",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Data object not found: /FNL\_SF\_Archive/projectx1/data[INVALID\_REQUEST\_INPUT]”  }  **File is not archived yet:**  This error is thrown if the file is not in ARCHIVED state.  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "REQUEST\_REJECTED",  "requestRejectReason": "FILE\_NOT\_ARCHIVED",  "message": "Object is not in archived state yet. It is in <IN\_PROGRESS\_TO\_TEMPORARY\_ARCHIVE> state",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Object is not in archived state yet. It is in IN\_PROGRESS\_TO\_TEMPORARY\_ARCHIVE state[REQUEST\_REJECTED]”  }  **Invalid Data transfer request:**  This error is thrown if an invalid Globus endpoint address is given  HTTP/1.1 500 Server ErrorContent-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "DATA\_TRANSFER\_ERROR",  "message": "Failed to activate endpoint: nihfnlcr#gridftp",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Failed to activate endpoint: nihfnlcr#gridftp[DATA\_TRANSFER\_ERROR]”  }  **Invalid Data transfer request:**  This error is thrown if Globus endpoint address(fileContainerId) or fileId is missing in the request  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid file location",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Invalid file location[INVALID\_REQUEST\_INPUT]”  } |

## Setting Entity (Collection or Data Object) Permissions

|  |  |
| --- | --- |
| Title | Set Permissions at metadata level |
| Description | Set users’ permissions for a list of entities (colelctions or data objects). The currently supported permissions are OWN, READ, WRITE and NONE.  Setting permissions of data file at Globus level/ data or object store level is not part of this API. This API would only set permissions at iRODS entity level so that users can access metadata and contribute to collections or data files.  These permissions can be associated with a user or a group. iRODS permissions are set by entity PATH. This entity can be a collection or a datafile. |
| URL | /hpc-server/acl |
| Method | POST |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params |  |
| Data Params | **JSON:**  {  "entityPermissionRequests": [  {  "path": "/FNL\_SF\_Archive/Project203/Dataset203",  "userPermissions": [  {  "userId": "user-id-1",  "permission": "OWN"  },  {  "userId": "user-id-2",  "permission": "READ"  }  ]  },  {  "path": "/FNL\_SF\_Archive/Project203/Dataset204",  "groupPermissions": [  {  "groupId": "group1",  "permission": "WRITE"  },  {  "groupId": "group2",  "permission": "NONE"  }  ]  }  ]  } |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0 |
| Error Response | **Missing path in the request:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Null or empty path in a permission request",  "stackTrace": …  }  **Missing User Permissions:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Null or empty user permission requests for path: /PI\_Lab/CSAS\_ID1/Run1/FlowCellId1/FastQ/ERR008979\_2.filt.fastq.gz",  "stackTrace":  }  **Invalid UserId:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Null or empty userId in a permission request for path: /FNL\_SF\_Archive/GlobusWorld2016",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Null or empty userId in a permission request for path: /FNL\_SF\_Archive/GlobusWorld2016[INVALID\_REQUEST\_INPUT] "  }  **Invalid GroupId:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Null or empty groupId in a permission request for path: /FNL\_SF\_Archive/GlobusWorld2016",  "stackTrace": "gov.nih.nci.hpc.exception.HpcException: Null or empty groupId in a permission request for path: /FNL\_SF\_Archive/GlobusWorld2016[INVALID\_REQUEST\_INPUT] "  }  **Authentication Failure:**  HTTP/1.1 401 Unauthorized  Content-Type: application/json  {"errorType":"REQUEST\_AUTHENTICATION\_FAILED","message":"Access Denied: LDAP authentication failed","stackTrace":"..”}  JSON:  {  "errorType": "REQUEST\_AUTHENTICATION\_FAILED",  "message": "Access Denied: LDAP authentication failed",  "stackTrace": "…"  } |

## Subscribe to notifications

|  |  |
| --- | --- |
| Title | Subscribe to event notifications |
| Description | Authorized users can subscribe or unsubscribe to following HPC DME event notifications:  DATA\_TRANSFER\_UPLOAD\_IN\_TEMPORARY\_ARCHIVE:  DATA\_TRANSFER\_UPLOAD\_ARCHIVED:  DATA\_TRANSFER\_UPLOAD\_FAILED:  DATA\_TRANSFER\_DOWNLOAD\_COMPLETED:  DATA\_TRANSFER\_DOWNLOAD\_FAILED:  USAGE\_SUMMARY\_REPORT:  USAGE\_SUMMARY\_BY\_WEEKLY\_REPORT:  Add/Update subscriptions should be included in the ‘addUpdateSubscriptions’ array, and delete subscriptions events should be included in the ‘deleteSubscriptions’ |
| URL | /hpc-server/notification/{nciUserId} |
| Method | POST |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params | nciUserId |
| Data Params | **JSON:**  {  "addUpdateSubscriptions": [  {  "eventType": "DATA\_TRANSFER\_UPLOAD\_ARCHIVED",  "notificationDeliveryMethods": ["EMAIL"]  },  {  "eventType": "DATA\_TRANSFER\_UPLOAD\_IN\_TEMPORARY\_ARCHIVE",  "notificationDeliveryMethods": ["EMAIL"]  },  {  "eventType": "DATA\_TRANSFER\_UPLOAD\_FAILED",  "notificationDeliveryMethods": ["EMAIL"]  },  {  "eventType": "DATA\_TRANSFER\_DOWNLOAD\_COMPLETED",  "notificationDeliveryMethods": ["EMAIL"]  },  {  "eventType": "DATA\_TRANSFER\_DOWNLOAD\_FAILED",  "notificationDeliveryMethods": ["EMAIL"]  },  {  "eventType": "USAGE\_SUMMARY\_REPORT",  "notificationDeliveryMethods": ["EMAIL"]  },  {  "eventType": "USAGE\_SUMMARY\_BY\_WEEKLY\_REPORT",  "notificationDeliveryMethods": ["EMAIL"]  }  ],  "deleteSubscriptions": [“DATA\_TRANSFER\_UPLOAD\_ARCHIVED”, “USAGE\_SUMMARY\_REPORT”]  } |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0 |
| Error Response | **Invaid notification type:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid add/update notification subscription request",  "stackTrace": ".."  }  **Authentication Failure:**  HTTP/1.1 401 Unauthorized  Content-Type: application/json  {"errorType":"REQUEST\_AUTHENTICATION\_FAILED","message":"Access Denied: LDAP authentication failed","stackTrace":"..”}  JSON:  {  "errorType": "REQUEST\_AUTHENTICATION\_FAILED",  "message": "Access Denied: LDAP authentication failed",  "stackTrace": "…"  } |

## Get notification subscription

|  |  |
| --- | --- |
| Title | Get notification subscriptions of a user |
| Description | Users can check what notifications they are subscribed to |
| URL | /hpc-server/notification/{nciUserId} |
| Method | GET |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params | nciUserId |
| Data Params |  |
| Success Response | HTTP/1.1 200 OK  Access-Control-Allow-Origin: \*  Access-Control-Allow-Methods: GET, POST, DELETE, PUT  HPC-API-Version: 1.0.0  Content-Type: application/json  Date: Thu, 25 Aug 2016 01:02:02 GMT  Transfer-Encoding: chunked  Server: Jetty(8.1.17.v20150415)  {"subscriptions": [  {  "eventType": "DATA\_TRANSFER\_DOWNLOAD\_COMPLETED",  "notificationDeliveryMethods": "EMAIL"  },  {  "eventType": "DATA\_TRANSFER\_UPLOAD\_IN\_TEMPORARY\_ARCHIVE",  "notificationDeliveryMethods": "EMAIL"  },  {  "eventType": "DATA\_TRANSFER\_UPLOAD\_ARCHIVED",  "notificationDeliveryMethods": "EMAIL"  },  {  "eventType": "DATA\_TRANSFER\_UPLOAD\_FAILED",  "notificationDeliveryMethods": "EMAIL"  },  {  "eventType": "DATA\_TRANSFER\_DOWNLOAD\_FAILED",  "notificationDeliveryMethods": "EMAIL"  }  ]} |
| Error Response | **Invaid notification user ID:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  Date: Thu, 25 Aug 2016 01:03:29 GMT  Transfer-Encoding: chunked  Server: Jetty(8.1.17.v20150415)  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid user: <user>  } |

## Save user query

|  |  |
| --- | --- |
| Title | Save a user query |
| Description | This API allows the user to save a compound query under a given name. The queries saved for the authenticated invoker and not shared with other users. |
| URL | /hpc-server/query/{queryName} |
| Method | POST |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params | queryName |
| Data Params | **JSON:**  {     "query":    {        "operator": "ANY",        "queries":       [                    {              "attribute": "ATTR 1",              "value": "VAL 1",              "operator": "EQUAL"           },                    {              "attribute": "ATTR 2",              "value": "VAL 2",              "operator": "LIKE"           }        ],        "compoundQueries":       [                    {              "operator": "ALL",              "queries":             [                                {                    "attribute": "ATTR 1",                    "value": "VAL 1",                    "operator": "EQUAL"                 },                                {                    "attribute": "ATTR 2",                    "value": "VAL 2",                    "operator": "LIKE"                 }              ]           },                    {              "operator": "ANY",              "queries":             [                                {                    "attribute": "ATTR 3",                    "value": "VAL 3",                    "operator": "NOT\_EQUAL"                 },                                {                    "attribute": "ATTR 4",                    "value": "VAL 4",                    "operator": "NUM\_LESS\_OR\_EQUAL"                 }              ]           }        ]     }  } |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0 |
| Error Response | **Invaid notification type:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid compound metadata query",  "stackTrace": ".."  }  **Authentication Failure:**  HTTP/1.1 401 Unauthorized  Content-Type: application/json  {"errorType":"REQUEST\_AUTHENTICATION\_FAILED","message":"Access Denied: LDAP authentication failed","stackTrace":"..”}  JSON:  {  "errorType": "REQUEST\_AUTHENTICATION\_FAILED",  "message": "Access Denied: LDAP authentication failed",  "stackTrace": "…"  } |

## Delete user query

|  |  |
| --- | --- |
| Title | Delete a user query |
| Description | Delete a user query |
| URL | /hpc-server/query/{queryName} |
| Method | DELETE |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params | queryName |
| Data Params |  |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0 |
| Error Response | **Invaid notification type:**  **Authentication Failure:**  HTTP/1.1 401 Unauthorized  Content-Type: application/json  {"errorType":"REQUEST\_AUTHENTICATION\_FAILED","message":"Access Denied: LDAP authentication failed","stackTrace":"..”}  JSON:  {  "errorType": "REQUEST\_AUTHENTICATION\_FAILED",  "message": "Access Denied: LDAP authentication failed",  "stackTrace": "…"  } |

## Get user queries

|  |  |
| --- | --- |
| Title | Get saved queries |
| Description | Return a list of all compound metadata queries that are saved for the invoker |
| URL | /hpc-server/query |
| Method | GET |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params |  |
| Data Params |  |
| Success Response | HTTP/1.1 200 OK  Access-Control-Allow-Origin: \*  Access-Control-Allow-Methods: GET, POST, DELETE, PUT  HPC-API-Version: 1.0.0  Content-Type: application/json  Date: Thu, 25 Aug 2016 01:02:02 GMT  Transfer-Encoding: chunked  Server: Jetty(8.1.17.v20150415)  {"queries": {     "name": "eran-query",     "compoundQuery":    {        "operator": "ANY",        "queries":       [                    {              "attribute": "ATTR 1",              "value": "VAL 1",              "operator": "EQUAL"           },                    {              "attribute": "ATTR 2",              "value": "VAL 2",              "operator": "LIKE"           }        ],        "compoundQueries":       [                    {              "operator": "ALL",              "queries":             [                                {                    "attribute": "ATTR 1",                    "value": "VAL 1",                    "operator": "EQUAL"                 },                                {                    "attribute": "ATTR 2",                    "value": "VAL 2",                    "operator": "LIKE"                 }              ]           },                    {              "operator": "ANY",              "queries":             [                                {                    "attribute": "ATTR 3",                    "value": "VAL 3",                    "operator": "NOT\_EQUAL"                 },                                {                    "attribute": "ATTR 4",                    "value": "VAL 4",                    "operator": "NUM\_LESS\_OR\_EQUAL"                 }              ]           }        ]     }  }} |
| Error Response | **Invaid notification user ID:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  Date: Thu, 25 Aug 2016 01:03:29 GMT  Transfer-Encoding: chunked  Server: Jetty(8.1.17.v20150415)  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid user: <user>  } |

## Generate Report

|  |  |
| --- | --- |
| Title | Generate usage summary report |
| Description | Authorized HPC DME users can generate following usage summary reports:  USAGE\_SUMMARY\_REPORT:  USAGE\_SUMMARY\_BY\_WEEKLY\_REPORT:  USAGE\_SUMMARY\_BY\_DOC\_REPORT:  USAGE\_SUMMARY\_BY\_DOC\_BY\_WEEKLY\_REPORT:  USAGE\_SUMMARY\_BY\_USER\_REPORT:  USAGE\_SUMMARY\_BY\_USER\_BY\_WEEKLY\_REPORT: |
| URL | /hpc-server/report |
| Method | POST |
| Acceptable request representation | application/json  application/xml |
| Available response representation | application/json  application/xml |
| URL Params |  |
| Data Params | **JSON:**  {  "type": "USAGE\_SUMMARY"  }  or  {  "type": "USAGE\_SUMMARY\_BY\_DATE\_RANGE",  "fromDate": "08/01/2016",  "toDate": "08/20/2016"  }  Or  {  "type": "USAGE\_SUMMARY\_BY\_DOC",  "doc": "FNLCR"  }  Or  {  "type": "USAGE\_SUMMARY\_BY\_DOC\_BY\_DATE\_RANGE",  "doc": "FNLCR",  "fromDate": "08/01/2016",  "toDate": "08/20/2016"  }  Or  {  "type": "USAGE\_SUMMARY\_BY\_USER",  "user": "konkapv"  }  Or  {  "type": "USAGE\_SUMMARY\_BY\_USER\_BY\_DATE\_RANGE",  "user": "konkapv",  "fromDate": "08/01/2016",  "toDate": "08/20/2016"  } |
| Success Response | HTTP/1.1 200 OK  Content-Type: application/json  HPC-API-Version: 1.0.0 |
| Error Response | **Invalid Report type:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Report type is missing",  "stackTrace": ".."  }  **Invalid Date format:**  HTTP/1.1 400 Bad Request  Content-Type: application/json  HPC-API-Version: 1.0.0  {  "errorType": "INVALID\_REQUEST\_INPUT",  "message": "Invalid fromDate format. Valid format is mm/dd/yyyy",  "stackTrace": ".."  }  **Authentication Failure:**  HTTP/1.1 401 Unauthorized  Content-Type: application/json  {"errorType":"REQUEST\_AUTHENTICATION\_FAILED","message":"Access Denied: LDAP authentication failed","stackTrace":"..”}  JSON:  {  "errorType": "REQUEST\_AUTHENTICATION\_FAILED",  "message": "Access Denied: LDAP authentication failed",  "stackTrace": "…"  } |

# APPENDIX A: HPC SERVER API GRAMMER

{

"grammars": {"xs.schema": [

{

"@attributeFormDefault": "unqualified",

"@elementFormDefault": "unqualified",

"@targetNamespace": "http://hpc.nci.nih.gov/dto/datamanagement",

"xs.import": [

{"@namespace": "http://hpc.nci.nih.gov/domain/datamanagement"},

{"@namespace": "http://hpc.nci.nih.gov/domain/metadata"},

{"@namespace": "http://hpc.nci.nih.gov/domain/datatransfer"}

],

"xs.element": [

{

"@name": "HpcCollectionListDTO",

"xs.complexType": {"xs.sequence": {"xs.element": {

"@maxOccurs": "unbounded",

"@minOccurs": "0",

"@name": "collections",

"@nillable": "true",

"@type": "tns:HpcCollectionDTO"

}}}

},

{

"@name": "HpcDataObjectDownloadDTO",

"xs.complexType": {"xs.sequence": {"xs.element": {

"@name": "destination",

"@type": "ns3:HpcFileLocation"

}}}

},

{

"@name": "HpcDataObjectListDTO",

"xs.complexType": {"xs.sequence": {"xs.element": {

"@maxOccurs": "unbounded",

"@minOccurs": "0",

"@name": "dataObjects",

"@nillable": "true",

"@type": "tns:HpcDataObjectDTO"

}}}

},

{

"@name": "HpcDataObjectRegistrationDTO",

"xs.complexType": {"xs.sequence": {"xs.element": [

{

"@name": "source",

"@type": "ns3:HpcFileLocation"

},

{

"@name": "filePath",

"@type": "xs:string"

},

{

"@maxOccurs": "unbounded",

"@name": "metadataEntries",

"@type": "ns2:HpcMetadataEntry"

}

]}}

},

{

"@name": "HpcEntityPermissionRequestDTO",

"xs.complexType": {"xs.sequence": {"xs.element": [

{

"@name": "path",

"@type": "xs:string"

},

{

"@maxOccurs": "unbounded",

"@minOccurs": "0",

"@name": "userPermissions",

"@nillable": "true",

"@type": "ns1:HpcUserPermission"

}

]}}

},

{

"@name": "HpcEntityPermissionResponseListDTO",

"xs.complexType": {"xs.sequence": {"xs.element": {

"@maxOccurs": "unbounded",

"@minOccurs": "0",

"@name": "entityPermissionResponses",

"@nillable": "true",

"@type": "tns:HpcEntityPermissionResponseDTO"

}}}

},

{

"@name": "HpcEntityPermissionRequestDTOs",

"xs.complexType": {"xs.sequence": {"xs.element": {

"@maxOccurs": "unbounded",

"@minOccurs": "0",

"@ref": "tns:HpcEntityPermissionRequestDTO"

}}}

}

],

"xs.complexType": [

{

"@name": "HpcEntityPermissionResponseDTO",

"xs.sequence": {"xs.element": [

{

"@name": "path",

"@type": "xs:string"

},

{

"@maxOccurs": "unbounded",

"@minOccurs": "0",

"@name": "userPermissionResponses",

"@nillable": "true",

"@type": "tns:HpcUserPermissionResponseDTO"

}

]}

},

{

"@name": "HpcUserPermissionResponseDTO",

"xs.sequence": {"xs.element": [

{

"@name": "userId",

"@type": "xs:string"

},

{

"@name": "result",

"@type": "xs:boolean"

},

{

"@name": "message",

"@type": "xs:string"

}

]}

},

{

"@name": "HpcCollectionDTO",

"xs.sequence": {"xs.element": [

{

"@name": "collection",

"@type": "ns1:HpcCollection"

},

{

"@maxOccurs": "unbounded",

"@name": "metadataEntries",

"@type": "ns2:HpcMetadataEntry"

}

]}

},

{

"@name": "HpcDataObjectDTO",

"xs.sequence": {"xs.element": [

{

"@name": "dataObject",

"@type": "ns1:HpcDataObject"

},

{

"@maxOccurs": "unbounded",

"@name": "metadataEntries",

"@type": "ns2:HpcMetadataEntry"

}

]}

}

]

},

{

"@attributeFormDefault": "unqualified",

"@elementFormDefault": "unqualified",

"@targetNamespace": "http://hpc.nci.nih.gov/domain/datamanagement",

"xs.complexType": [

{

"@name": "HpcCollection",

"xs.sequence": {"xs.element": [

{

"@name": "collectionId",

"@type": "xs:int"

},

{

"@name": "collectionName",

"@type": "xs:string"

},

{

"@name": "absolutePath",

"@type": "xs:string"

},

{

"@name": "collectionParentName",

"@type": "xs:string"

},

{

"@name": "collectionOwnerName",

"@type": "xs:string"

},

{

"@name": "collectionOwnerZone",

"@type": "xs:string"

},

{

"@name": "collectionMapId",

"@type": "xs:string"

},

{

"@name": "collectionInheritance",

"@type": "xs:string"

},

{

"@name": "comments",

"@type": "xs:string"

},

{

"@name": "info1",

"@type": "xs:string"

},

{

"@name": "info2",

"@type": "xs:string"

},

{

"@name": "createdAt",

"@type": "xs:date"

},

{

"@name": "modifiedAt",

"@type": "xs:date"

},

{

"@name": "specColType",

"@type": "xs:string"

}

]}

},

{

"@name": "HpcDataObject",

"xs.sequence": {"xs.element": [

{

"@name": "id",

"@type": "xs:int"

},

{

"@name": "collectionId",

"@type": "xs:int"

},

{

"@name": "dataName",

"@type": "xs:string"

},

{

"@name": "collectionName",

"@type": "xs:string"

},

{

"@name": "absolutePath",

"@type": "xs:string"

},

{

"@name": "dataReplicationNumber",

"@type": "xs:int"

},

{

"@name": "dataVersion",

"@type": "xs:int"

},

{

"@name": "dataSize",

"@type": "xs:long"

},

{

"@name": "dataTypeName",

"@type": "xs:string"

},

{

"@name": "resourceGroupName",

"@type": "xs:string"

},

{

"@name": "resourceName",

"@type": "xs:string"

},

{

"@name": "dataPath",

"@type": "xs:string"

},

{

"@name": "dataOwnerName",

"@type": "xs:string"

},

{

"@name": "dataOwnerZone",

"@type": "xs:string"

},

{

"@name": "replicationStatus",

"@type": "xs:string"

},

{

"@name": "dataStatus",

"@type": "xs:string"

},

{

"@name": "checksum",

"@type": "xs:string"

},

{

"@name": "expiry",

"@type": "xs:string"

},

{

"@name": "dataMapId",

"@type": "xs:int"

},

{

"@name": "comments",

"@type": "xs:string"

},

{

"@name": "createdAt",

"@type": "xs:date"

},

{

"@name": "updatedAt",

"@type": "xs:date"

},

{

"@name": "specColType",

"@type": "xs:string"

}

]}

},

{

"@name": "HpcUserPermission",

"xs.sequence": {"xs.element": [

{

"@name": "userId",

"@type": "xs:string"

},

{

"@name": "permission",

"@type": "xs:string"

}

]}

}

]

},

{

"@attributeFormDefault": "unqualified",

"@elementFormDefault": "unqualified",

"@targetNamespace": "http://hpc.nci.nih.gov/dto/user",

"xs.import": {"@namespace": "http://hpc.nci.nih.gov/domain/user"},

"xs.element": [

{

"@name": "HpcAuthenticationRequestDTO",

"xs.complexType": {"xs.sequence": {"xs.element": [

{

"@name": "userName",

"@type": "xs:string"

},

{

"@name": "password",

"@type": "xs:string"

}

]}}

},

{

"@name": "HpcAuthenticationResponseDTO",

"xs.complexType": {"xs.sequence": {"xs.element": [

{

"@name": "authenticated",

"@type": "xs:boolean"

},

{

"@name": "userRole",

"@type": "xs:string"

}

]}}

},

{

"@name": "HpcUserDTO",

"xs.complexType": {"xs.sequence": {"xs.element": [

{

"@name": "nciAccount",

"@type": "ns1:HpcNciAccount"

},

{

"@name": "dataTransferAccount",

"@type": "ns1:HpcIntegratedSystemAccount"

},

{

"@name": "dataManagementAccount",

"@type": "ns1:HpcIntegratedSystemAccount"

},

{

"@name": "dataManagementUserType",

"@type": "xs:string"

}

]}}

}

]

},

{

"@attributeFormDefault": "unqualified",

"@elementFormDefault": "unqualified",

"@targetNamespace": "http://hpc.nci.nih.gov/domain/metadata",

"xs.complexType": [

{

"@name": "HpcMetadataEntry",

"xs.sequence": {"xs.element": [

{

"@name": "attribute",

"@type": "xs:string"

},

{

"@name": "value",

"@type": "xs:string"

},

{

"@name": "unit",

"@type": "xs:string"

}

]}

},

{

"@name": "HpcMetadataValidationRule",

"xs.sequence": {"xs.element": [

{

"@name": "attribute",

"@type": "xs:string"

},

{

"@name": "mandatory",

"@type": "xs:boolean"

},

{

"@maxOccurs": "unbounded",

"@minOccurs": "0",

"@name": "validValues",

"@nillable": "true",

"@type": "xs:string"

},

{

"@name": "defaultValue",

"@type": "xs:string"

},

{

"@name": "defaultUnit",

"@type": "xs:string"

},

{

"@name": "collectionType",

"@type": "xs:string"

},

{

"@name": "ruleEnabled",

"@type": "xs:boolean"

},

{

"@name": "DOC",

"@type": "xs:string"

}

]}

},

{

"@name": "HpcMetadataQuery",

"xs.sequence": {"xs.element": [

{

"@name": "attribute",

"@type": "xs:string"

},

{

"@name": "value",

"@type": "xs:string"

},

{

"@name": "operator",

"@type": "xs:string"

}

]}

}

],

"xs.element": [

{

"@name": "HpcMetadataEntry",

"@type": "HpcMetadataEntry"

},

{

"@name": "HpcMetadataEntrys",

"xs.complexType": {"xs.sequence": {"xs.element": {

"@maxOccurs": "unbounded",

"@minOccurs": "0",

"@ref": "HpcMetadataEntry"

}}}

}

]

},

{

"@attributeFormDefault": "unqualified",

"@elementFormDefault": "unqualified",

"@targetNamespace": "http://hpc.nci.nih.gov/domain/datatransfer",

"xs.complexType": {

"@name": "HpcFileLocation",

"xs.sequence": {"xs.element": [

{

"@name": "endpoint",

"@type": "xs:string"

},

{

"@name": "path",

"@type": "xs:string"

}

]}

}

},

{

"@attributeFormDefault": "unqualified",

"@elementFormDefault": "unqualified",

"@targetNamespace": "http://hpc.nci.nih.gov/domain/user",

"xs.complexType": [

{

"@name": "HpcNciAccount",

"xs.sequence": {"xs.element": [

{

"@name": "userId",

"@type": "xs:string"

},

{

"@name": "firstName",

"@type": "xs:string"

},

{

"@name": "lastName",

"@type": "xs:string"

}

]}

},

{

"@name": "HpcIntegratedSystemAccount",

"xs.sequence": {"xs.element": [

{

"@name": "username",

"@type": "xs:string"

},

{

"@name": "password",

"@type": "xs:string"

},

{

"@name": "integratedSystem",

"@type": "xs:string"

}

]}

}

],

"xs.simpleType": {

"@name": "HpcIntegratedSystem",

"xs.restriction": {

"@base": "xs:string",

"xs.enumeration": [

{"@value": "GLOBUS"},

{"@value": "IRODS"}

]

}

}

}

]},

"resources": {

"@base": "http://fr-s-hpcdm-gp-d.ncifcrf.gov:7737/hpc-server",

"resource": [

{

"@path": "/",

"resource": [

{

"@path": "user",

"method": {

"@name": "PUT",

"request": {"representation": [

{

"@element": "HpcUserDTO",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]},

"response": {"representation": {"@mediaType": "\*/\*"}}

}

},

{

"@path": "user/authenticate",

"method": {

"@name": "POST",

"request": {"representation": [

{

"@element": "HpcAuthenticationRequestDTO",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]},

"response": {"representation": {"@mediaType": "\*/\*"}}

}

},

{

"@path": "user/{nciUserId}",

"param": {

"@name": "nciUserId",

"@style": "template",

"@type": "string"

},

"method": {

"@name": "GET",

"request": "",

"response": {"representation": [

{

"@element": "Response",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]}

}

}

]

},

{

"@path": "/",

"resource": [

{

"@path": "acl",

"method": {

"@name": "POST",

"request": {"representation": {"@mediaType": "\*/\*"}},

"response": {"representation": [

{

"@element": "Response",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]}

}

},

{

"@path": "collection",

"method": {

"@name": "GET",

"request": {"param": {

"@name": "metadataQuery",

"@repeating": "true",

"@style": "query"

}},

"response": {"representation": [

{

"@element": "Response",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]}

}

},

{

"@path": "collection/{path:.\*}",

"param": {

"@name": "path",

"@style": "template",

"@type": "string"

},

"method": [

{

"@name": "GET",

"request": "",

"response": {"representation": [

{

"@element": "Response",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]}

},

{

"@name": "PUT",

"request": {"representation": [

{

"@element": "HpcMetadataEntry",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]},

"response": {"representation": [

{

"@element": "Response",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]}

}

]

},

{

"@path": "dataObject",

"method": {

"@name": "GET",

"request": {"param": {

"@name": "metadataQuery",

"@repeating": "true",

"@style": "query"

}},

"response": {"representation": [

{

"@element": "Response",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]}

}

},

{

"@path": "dataObject/{path:.\*}",

"param": {

"@name": "path",

"@style": "template",

"@type": "string"

},

"method": [

{

"@name": "GET",

"request": "",

"response": {"representation": [

{

"@element": "Response",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]}

},

{

"@name": "PUT",

"request": {"representation": [

{

"@element": "HpcDataObjectRegistrationDTO",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]},

"response": {"representation": [

{

"@element": "Response",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]}

}

]

},

{

"@path": "dataObject/{path:.\*}/download",

"param": {

"@name": "path",

"@style": "template",

"@type": "string"

},

"method": {

"@name": "POST",

"request": {"representation": [

{

"@element": "HpcDataObjectDownloadDTO",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]},

"response": {"representation": [

{

"@element": "Response",

"@mediaType": "application/json"

},

{"@mediaType": "application/xml"}

]}

}

}

]

}

]

}

}