**Overview and Assessment**

**of**

**iRODS Integration**

**10/05/2015**

Table of Contents

[Overview 4](#_Toc436387782)

[1.1 HPC requirements 4](#_Toc436387783)

[1.2 Overview of iRODS 7](#_Toc436387784)

[1.3 Overview of iRODS Schema 9](#_Toc436387785)

[1.4 iRODS and HPC DME Overlay 10](#_Toc436387786)

[1.5 HPC Requirements In iRODS 12](#_Toc436387787)

[1.5.1 Data Objects within HPC and iRODS 13](#_Toc436387788)

[1.5.2 User Accounts 17](#_Toc436387789)

[1.5.3 User Authentication 17](#_Toc436387790)

[1.5.4 Data Registration with metadata 18](#_Toc436387791)

[1.5.5 Data Transfer 19](#_Toc436387792)

[1.5.6 Metadata 20](#_Toc436387793)

[1.5.7 Search for Data by metadata 21](#_Toc436387794)

[1.5.8 Roles and Authorizations 22](#_Toc436387795)

[1.5.9 Reporting 26](#_Toc436387796)

[1.5.10 Feature Contrast 26](#_Toc436387797)

# Overview

## HPC requirements

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. HPC DME provides capabilities for storing, managing, transferring and sharing large data across different systems securely and efficiently. The HPC data management environment provides a high-reliability storage model for underlying datasets including a dataset registration system, and an API for transfer of large datasets with no-loss of data. The dataset registration system associates a label with a given managed dataset and captures extensible metadata for the managed dataset. The basic features of HPC data management system help users in registering and uploading their data to the HPC DM archive storage and managing it. HPC DMEDME 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.

The HPC data management environment provides a number of application programming interfaces (APIs) to operate and interact with it. By default, the HPC data management system integrates with Globus platform and uses it to perform data transfer tasks. Each data file is stored along with its system generated and user defined metadata associated with it. The associated metadata can be used as search criteria to identify dataset(s). The HPC data management system is a data file centric system, where a dataset is defined as an individual file or a set of files, referred to as a ‘Dataset’, and forms the basic unit of all the functionalities. A dataset can have one or more data files. Another important component of the HPC system is a ‘Project’. Projects can be of different types for example, the dataset can be linked to a sequencing project, imaging study or an analysis project etc., or a project can be an entity (umbrella project) indicating a scientific ‘study’.

HPC DME collects three kinds of metadata related to a dataset or Project, namely, system generated, administrative and domain specific. The administrative metadata is the required set of information which needs to be submitted at the time of registration with HPC DME. The domain specific metadata is more related to the field of study and is also extensible. The metadata can also be updated by authorized users.

HPC UI reference implementation provides an easy to interact with Web interface that works with HPC Server API. HPC Server API is a RESTful implementation that can be accessed via any client interface to register, transfer and manage dataset and its associations.

Server API can be categorized into following sections as of now (Oct1st). The following given capabilities are evolving iteratively through Agile scrum methodology. HPC DME Server API specifications will be revised as more functionality is added.

* User
  + Enroll user into HPC DME
  + Authenticate user with NCI LDAP
* Dataset
  + Register dataset along with metadata into HPC DME
  + Find Dataset by Id
  + Find Dataset by name
  + Find Dataset by Registrar Id
  + Find Dataset by Primary Investigator Id
  + Find Dataset by Project Id
  + Find Dataset by Primary metadata
* Project
  + Register project with its primary metadata into HPC DME
  + Find Project by Id
  + Find Project by Registrar Id
  + Find Project by Primary Investigator Id

HPC UI reference implementation provides an easy to interact with Web interface that works with HPC Server API. HPC Server APIs are a RESTful implementation that can be accessed via any client interface to register, transfer and manage dataset and its associations. The basic features of HPC DME user interface helps users in uploading their data to the archive storage system and managing it. The present features are listed below:

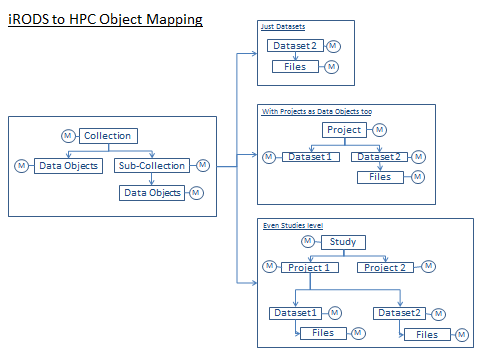
* NCI User authentication
* Dataset registration
* Project registration
* Dataset search
* Project Search

HPC DME collects and stores system generated and user-defined metadata. The Metadata is grouped into primary and secondary metadata. The primary metadata is collected at the time of dataset or project registration with HPC DME and represents the minimal information required about the data which will help in identifying the dataset precisely. The submission of primary metadata is mandated by HPC DME. However, user is provided options to inform of unavailability of the required information. The secondary metadata is domain specific and defined by the user based on their projects and datasets. HPC DME provides capabilities to the users to be able to add new metadata variables to the primary and secondary metadata and also be able to update them when needed.

## Overview of iRODS

iRODS is open-source, data management software that lets users access, manage, and share data across any type or number of storage systems located anywhere, while maintaining redundancy and security, and exercise precise control over their data with extensible rules that ensure the data is archived, described, and replicated in accordance with their needs and schedule. iRODS empowers users by supporting virtualization, which provides a one stop shop for all data regardless of the heterogeneity of storage devices, data discovery through the use of descriptive metadata, workflow automation, and data sharing between collaborating or distributed teams. iRODS is a layer that sits above the file systems that contain data, and below domain-specific applications. Because iRODS has a plugin framework and is technology-agnostic, it provides insulation from vendor lock-in. System administrators can slide iRODS on top of an existing heterogeneous data infrastructure and construct a flexible data grid. As middleware, iRODS allows administrators to track and control access to the data under their care.

In iRODS, the term Data Object refers to the logical representation of data that maps to one or more physical instances of the data at rest in storage resources. Data objects are organized into hierarchical Collections—the logical representations of physical containers, similar to directories or folders that are found in a file system. A Collection can have sub-collections, and hence provides a hierarchical structure. This extensible property of iRODS enables overlay of various different hierarchical structures or folder structures to be represented easily. The image below represents how different hierarchical folder structures can be represented using iRODS data objects.

****

New policies could be incorporated into iRODS by making rules and implementing them via microservices. iRODS provides microservices which are small, well-defined procedures that perform a server-side task and are either compiled into the iRODS server code or packaged independently as a shared object. Rules invoke Microservices to implement data management policies.

## Overview of iRODS Schema

Each iRODS deployment—or Zone—is composed of an iRODS Metadata Catalog (iCAT), an iCAT-Enabled Server (IES), and optional Resource Servers. The iCAT is a relational database that holds all the information about your data, users, and zone that the iRODS servers—IES and any resource servers—need to facilitate the management and sharing of your data. The iCAT contains the information about

• the zone for the purposes of sharing across zones,

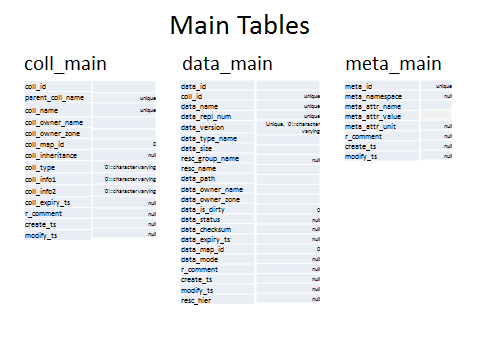
• data and their metadata,

• the virtual file system,

• resource configuration, and

• user information.

The iRODS data elements are grouped into several tables. The most important tables which map to HPC requirements are the tables representing Collections, Data Objects and the Metadata. These tables are the coll\_main, data\_main and meta\_main. These tables are shown in the image below. The image also shows the fields which can have a ‘Null’ value and which ones can’t. The collections and the data tables are connected to the metadata table by the map\_id column. Metadata may be attached to files, users, groups, collections (iRODS equivalent of sub-directories), and resources.

****

## iRODS and HPC DME Overlay

|  |  |
| --- | --- |
| **HPC DME Features** | **iRODS Features** |
| HPC DME data structure provisions allow creation of Projects and Datasets, each associated with their metadata. | iRODS allow creation of Collections and Data Objects, each with associated metadata. |
| HPC DME uses MongoDb for storing its metadata. The metadata has reference to the objects in the object store. | Each iRODS zone contains an iCAT-enabled resource server, which uses a relational database to organize the content of the zone and to maintain iRODS metadata. |
| HPC DME stores the metadata as Key-Value Pair. | The iCAT server 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. |
| HPC DME permits both System generated and User defined metadata. | Metadata may be user-defined or applied automatically. |
| HPC DME utilizes metadata for data discovery. | iRODS also utilizes metadata for data discovery. |
| HPC DME has a rich collection of search APIs which utilize metadata to search for datasets as well as Projects. | iRODS metadata can be searched as well. A simple way to search is using the iRODS imeta command. More complex queries can be generated using a subset of SQL operations issued through the iquest command. |
| HPC DME requirements include secure sharing of data between users and user groups. | iRODS is built to support federation, so that the preservation environment may share data with other institutions or organizations while remaining under your own audit and policy control. |
| HPC DME has a defined list of business rules for access control, data creation, data transfer, metadata creation, metadata extraction, metadata storage, backup, data migration etc. | iRODS has a rule engine which can run organizational policies or business rules, which can help customize and automate related data management tasks. |

## HPC Requirements In iRODS

This section describes the compare and contrast analysis of HPC requirements and iRODS functionalities. most common use cases of HPC DME and if and how they can be implemented in iRODS.

### Data Objects within HPC and iRODS

#### HPC Structure

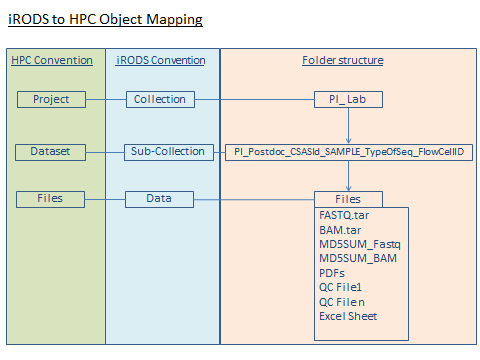
The HPC groups the data objects into datasets which are defined as an individual data file or a set of data files, registered together as one unit. A collection of datasets representing a study, experiment or a scientific procedure during which one or more datasets are generated and grouped is referred to as a Project. HPC DME stores and associates metadata to any registered dataset or project at different levels of data life cycle.

#### iRODS Structure

iRODS data elements and structure are flexible and extensible. They can be structured to represent most common hierarchical data structures. iRODS refers to the data files as data objects and groups them into hierarchical and logical representations of physical containers, similar to directories or folders that are found in a file system called Collections.

#### Feasibility

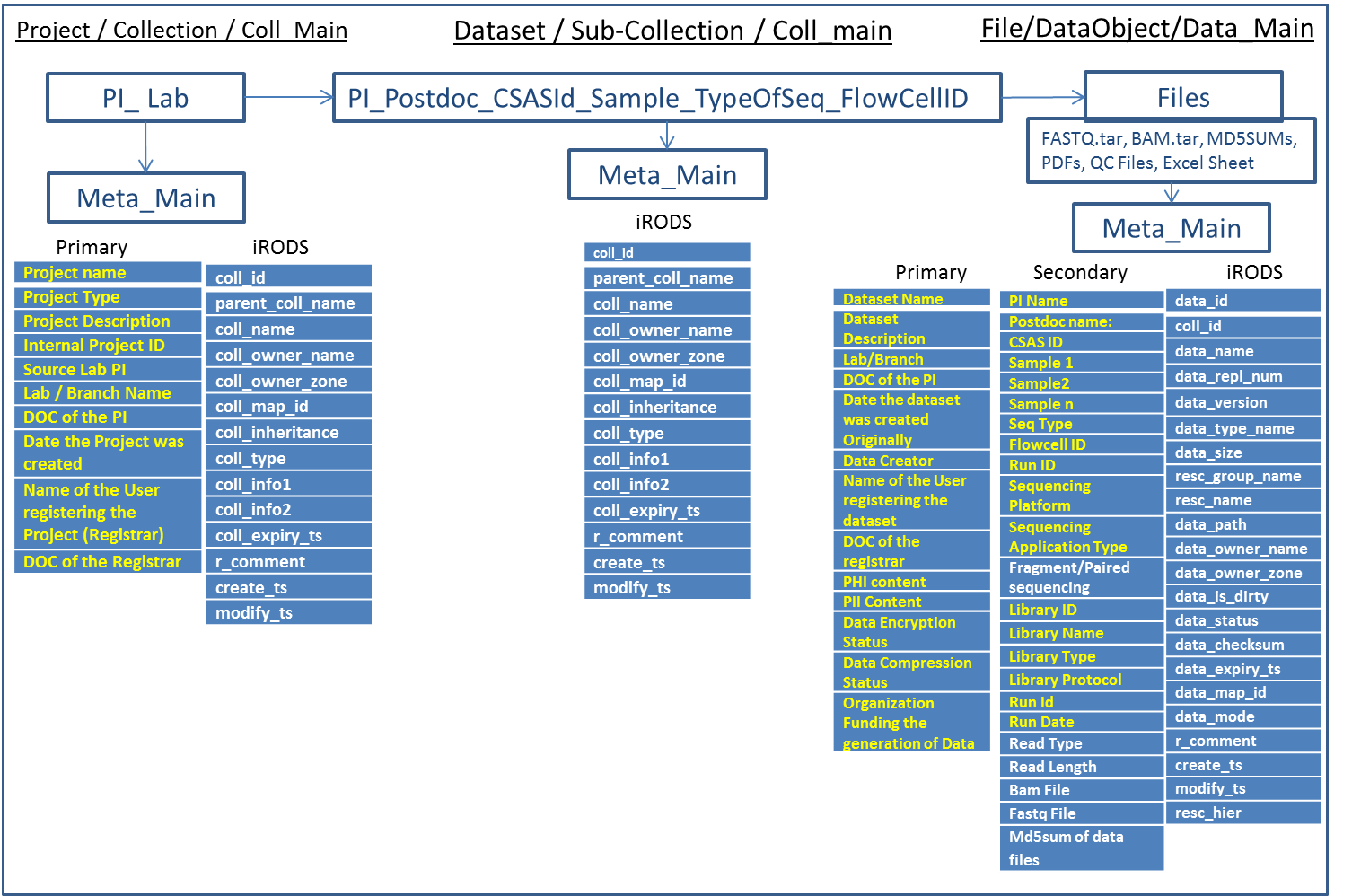
HPC Data objects and collections are well complemented by the iRODS data structures and can match the HPC convention in different possible ways. The image below shows the overlay of HPC DME data objects to iRODS data objects and an example of users storing their scientific data on other file systems.



A typical HPC use case (NGS domain) is described below:

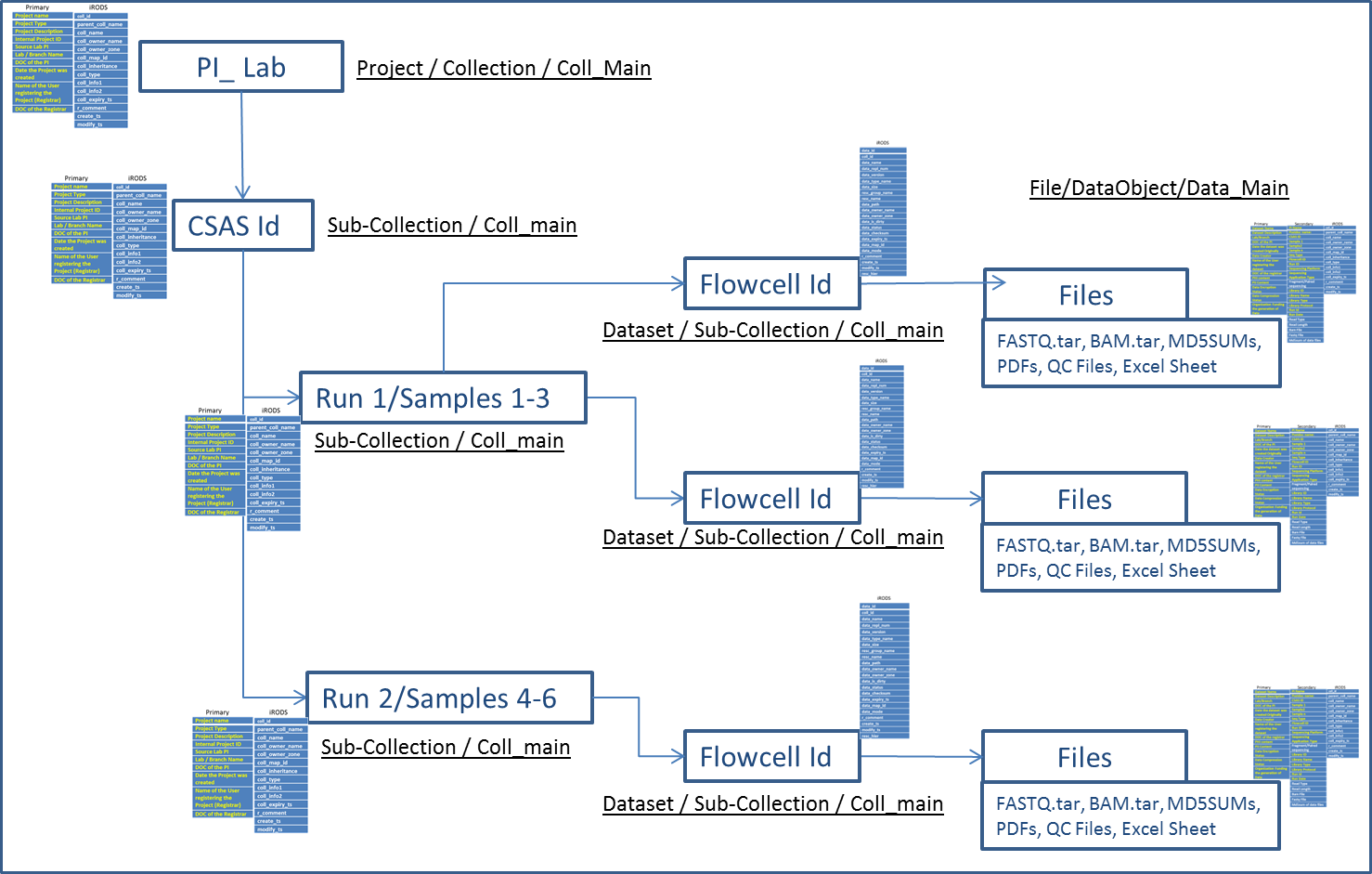
* *A postdoc, from a PI lab, is performing exome sequencing on a project assigned to him (CSAS ID) with 6 samples. She generates data sets using Illumina technology for DHS and ChIP-Seq experiments. This data needs to be stored and annotated such that she would be able to find the data quickly and easily whenever needed.*

The user could store this kind of data and associated metadata in HPC DME and iRODS. The main folder, is a Project in HPC terms and a Collection in IRODS terms. This project or parent collection can have several datasets and will overlay the sub collections of iRODS. Each of these datasets will then have data objects or individual files under them. Each of these components will have their associated metadata linked and stored in the database such that the user can search for particular datasets or files under this project. For ex: search for files generated under a CSAS id.

****

iRODS can also support a deeper hierarchical folder structure/ object storage structure based on user requirements. The image below represents an example, where the users might have more than one level in the hierarchical folder structure or object storage structure. The users can represents the project under multiple hierarchical groups based on their process or policy management.

iRODS can reinforce multi-level hierarchical structure for storing data files with associated metadata. Each of the collections and Sub-collections can have associations to relevant metadata such as parent collection and other user defined metadata. The actual data objects/Files can have the administrative and domain specific (HPC primary and secondary) metadata.



### User Accounts

#### HPC Requirements

All NIH users can log into HPC DME.

#### iRODS Implementation

#### Gaps

### User Authentication

#### HPC Requirements

* HPC DME follows NCI LDAP and allows any user with NIH account to log in via UI or access the system APIs.

#### iRODS Implementation

* By default, iRODS uses a secure password system for user authentication. The user passwords are scrambled and stored in the iCAT database.
* Additionally, iRODS supports user authentication via PAM (Pluggable Authentication Modules), which can be configured to support many things, including the LDAP authentication system.

#### Gaps

* With iRODS, PAM and SSL have been configured 'available' out of the box, but there is still some setup required to configure an installation to communicate with your external authentication server of choice.

### Data Registration with metadata

#### HPC Requirements

* HPC DME requirements need users to register their data with HCP for storing their data in HPC object storage.
* The HPC DME registration process provides users with an interface where they could upload their data files or collections of files to store into the HPC object storage with a given name and description.
* HPC registration interface also provides to collect metadata associated to the data objects at the time of registration.
* The main requirement feature of HPC DME is that it can mandate a set of required metadata to be collected at the time of registration. The required metadata associated with a data object is comprised of metadata attributes required by the business policies and also attributes defined by the users.
* This helps the users in making sure the data is stored in HPC along with the important information attached which could represent the status, worth and credibility of the data for future reuse.

#### iRODS Implementation

* In iRODS, users can upload their data without any metadata attached. Registration necessarily means upload of individual data files or folders.
* By default adding metadata to the files is the users choice.

#### Gaps

* In iRODS, by default here is no way to make sure that the uploaded data files have metadata associated to them. It needs customization to add this feature.
* HPC mandating required metadata which could
  + help in searching and grouping the data files using metadata attributes
  + formulating rich queries
  + help in analyzing the data accurately
  + represent the value of the data file/ dataset over a period of time
  + help in generating reports based on domain specific attributes

### Data Transfer

#### HPC Requirements

* HPC integrates with Globus platform to perform any kind of data transfers.
* Globus handles the data transfer, allowing application users to easily start and manage transfers between endpoints, while automatically tuning parameters to maximize bandwidth usage, managing security configurations, providing automatic fault recovery, and notifying users of completion and problems.
* A Globus “endpoint” is one of the two file transfer locations – either the source or the destination – between which files can move. Once a resource (such as a server, cluster, storage system, laptop, or other system) is defined as an endpoint, it will be available to authorized users who can transfer files to or from this endpoint.
* Globus uses GridFTP for more reliable and high-performance file transfer, and queues file transfers to be performed asynchronously in the background. This provides high-performance and reliable upload or download mechanism for large (or small) quantities of data. Users can therefore move data not only between the application and the user, but also between any other Globus endpoints. This is advantageous in research domains, as application users often need to move data between different locations during their normal data management lifecycle.

#### iRODS Implementation

#### Gaps

### Metadata

#### HPC Requirements

* HPC requirements define the metadata into required and not-required metadata. The required metadata is collected along with the data from the users. The not-required metadata is optional and can be defined by the users.
* HPC DME also collects system generated metadata associated to the data and the user, such as the date of registration, the user registering the data etc. at the time of registration.
* HPC requirements provide flexibility to be able to add user defined optional metadata associated with each project, dataset or files. Metadata attributes can be added to data objects and collections and add values to it later.
* HPC DME provides capability to configure default values for metadata attributes which unless specified by the user will retain the default value.
* HPC DME provides capability to configure list values for metadata attributes which would have a list of values to choose from.

#### iRODS Implementation

* In iRODS, metadata is stored as strings in the form of attribute-value-unit (AVU) triples. Attribute and value are required, but units are not required and can be empty strings.
* Besides the user defined metadata iRODS generates metadata about the data at the time of data transfer.

#### Gaps

* iRODS does not provide any default values for metadata attributes so if the users do not have a value for the attribute at that time they will not be able to have a default value nor will be able to add the metadata attribute as iRODS does not allow creating a metadata attribute without a value.
* iRODS does not provide any list values for metadata attributes. Metadata attributes which can have a set of list values to choose from can be user friendly as well as be a means to use controlled vocabulary.

### Search for Data by metadata

#### HPC Requirements

* HPC DME facilitates search for data files and collections of files (datasets and projects) using system-generated and user-defined metadata as search criteria.
* HPC DME provides capabilities to perform complex queries using multiple metadata attributes as criteria.
* HPC DME allows searches using wild character.

#### iRODS Implementation

* iRODS has different search capabilities via different access interfaces.
* iRODS UI has limited search capabilities where users can only serach by tags generated by them for each data object or collection.
* iRODS command line interface (icommands) allows search for data objects via ‘ilocate’ which searches for data objects in local zone with full paths with components which match the provided search pattern with ‘%’ as wildcard.
* iRODS command line interface (icommands) also allows search for data objects via ‘imeta qu’ using metadata as search criteria.
* iRODS command line interface (icommands) also allows search for data objects via ‘iquest’ which is a search using a pre-defined sql query.

#### Gaps

### Roles and Authorizations

#### HPC Requirements

* HPC DME shall have roles which would provide enough flexibility and control to individual groups and users at the same time maintaining control over data and its integrity.
* HPC DME proses the following roles:
  + HPC System Administrator
  + Super User
  + Owner
  + User
* Each role could be granted the following possible authorizations
  + Create data object
  + View data object
  + Share data object
  + Stop Sharing data object
  + Renew Sharing data object
  + Update data object
  + Create Metadata
  + Update Metadata
  + Delete Metadata

#### iRODS Implementation

* iRODS primarily provides three primary roles,
  + Admin
  + User
  + Groupadmin
* iRODS provides the following access types:
  + Own
  + Read
  + Write
  + Null
* An iRODS admin can control and modify user access and permissions on different objects and collections.
* A group admin can create users, groups and add or delete users from groups.
* A user, as an owner of collections or objects can assign different access types to different users on the owned collections or objects.

#### Gaps

* iRODS access types do not differentiate between access to data objects and metadata. Users who have read or write access on the data objects or collections have the same access to the associated metadata.
* In iRODS, the user who stores the collections or objects, is the owner of those collections or objects. This might not be the usecase all users might want to adopt.

The following table shows the Access types from HPC requirements and iRODS and a way to integrate

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **AccessType** | **iRODS Access\_Type** | **Roles and Default Authorizarion** | | |
| iRods Role |  |  |  | **rodsadmin** | **groupadmin** | **rodsuser** |
| HPC Role | User |  |  | Sys Admin | Super User | User (Default) |
|  |  | Create User |  | Yes | No | No |
|  |  | Update User |  | Yes | No | No |
|  |  | Update User Access | administer object | Yes | Yes | No |
|  |  | Delete User |  | Yes | No | No |
|  | Group |  |  |  |  |  |
|  |  | Create Group |  | Yes | Yes | No |
|  |  | Update Group |  | Yes | Yes | No |
|  |  | Delete Group |  | Yes | Yes | No |
|  | Objects |  |  |  |  |  |
|  | Project | Create Project | Own/Write | Yes | Yes | No |
|  |  | View Project | Own/Read | Yes | Yes | Yes (own, shared,public) |
|  |  | Update Project | Own/Write | Yes | Yes | No |
|  |  | Share Project (Create,Read and Update) |  | Yes | Yes | No |
|  |  | Export Metadata |  | Yes | Yes | Yes |
|  |  | Create Metadata | Own/Write | Yes | Yes | No |
|  |  | View Metadata | Own/Read | Yes | Yes | Yes |
|  |  | Update Required Metadata | Own/Write | Yes | Yes | No |
|  |  | Update Not-Required Metadata | Own/Write | Yes | Yes | No |
|  |  | Delete Metadata | Own/Delete | Yes | Yes | No |
|  |  |  |  |  |  | No |
|  | Dataset | Create Dataset | Own/Write | Yes | Yes | No |
|  |  | View Dataset | Own/Read | Yes | Yes | Yes (own, shared,public) |
|  |  | Download Dataset |  | Yes | Yes | No |
|  |  | Update Dataset | Own/Write | Yes | Yes | No |
|  |  | Share Dataset (Create,Read and Update) |  | Yes | Yes | No |
|  |  | Export Metadata |  | Yes | Yes | No |
|  |  | Create Metadata | Own/Write | Yes | Yes | No |
|  |  | View Metadata | Own/Read | Yes | Yes | yes |
|  |  | Update Required Metadata | Own/Write | Yes | Yes | No |
|  |  | Update Not-Required Metadata | Own/Write | Yes | Yes | No |
|  |  | Delete Metadata | Own | Yes | Yes | No |
|  |  |  |  |  |  | No |
|  | Data File | Create Data File | Own/Write | Yes | Yes | No |
|  |  | View Data File | Own/Read | Yes | Yes | Yes |
|  |  | Download Data File |  | Yes | Yes | No |
|  |  | Update Data File | Own/Write | Yes | Yes | No |
|  |  | Share Data File (Create,Read and Update) |  | Yes | Yes | No |
|  |  | Export Metadata |  | Yes | Yes | No |
|  |  | Create Metadata | Own/Write | Yes | Yes | No |
|  |  | View Metadata | Own/Read | Yes | Yes | Yes |
|  |  | Update Required Metadata | Own/Write | Yes | Yes | No |
|  |  | Update Not-Required Metadata | Own/Write | Yes | Yes | No |
|  |  | Delete Metadata | Own | Yes | Yes | No |

### Reporting

#### HPC Requirements

* HPC DME requirements need some reporting capabilities to provide system generated utilization and logistics reports as required or periodically.
* These reports generated capture information on storage access and utilization by each user or group/s and the metadata coverage of data objects.

#### iRODS Implementation

* iRODS presents limited reporting functionality which is limited to only Zone reports and error reports.

#### Gaps

### Feature Contrast

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **HPC** | **iRODS** | **Globus** |
| **User Accounts** | Possible | Possible |  |
| **User Authentication** | Possible | Possible |  |
| **Data Registration** | Possible with required metadata | Plain data upload |  |
| **Data Transfer** | | | |
| Data Transfer | Asynchronous  (using Globus) | Synchronous | Asynchronous |
| Transfer endpoint? |  |  |  |
| **Metadata** | | | |
| Metadata Versioning | Possible | Not Possible (Internally, the system removes the old AVU from the object and creates a new one (ensuring consistency), and performs a single 'commit' if all is valid.) |  |
| Mandate required metadata before data transfer | Possible | Not Possible |  |
| Optional Metadata (Attribute with Null as value) | Possible | Not Possible (Cannot create an attribute with null as value) |  |
| Provide Metadata default values and List values | Possible | Not Possible |  |
| Metadata Authorization | Can be implemented | Not possible |  |
| **Search Criteria** | | | |
| Search | Search using system and user-defined metadata | -ilocate --Search data objects in local zone with full paths with components which match the provided search pattern with ‘%’ as wildcard  -imeta qu – Search metadata  -iquest – Search using a pre-defined sql query. |  |
| **Authorization** | | | |
| Authorization and Roles | Can be implemented | Has three User types (Admin, User and Groupadmin) and four access types (Own, Write, Read, Null) |  |
| **Reports** | | | |
| Utilization Report | Can be implemented – Multiple reports based on data and metadata. | Only Zone reports and error reports. |  |