*HPC Data MANAGEMENT*

adminISTRATION guide

Version *1.1*

*05/05/2017*

**Version History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version Number** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Description of Change** |
| 1.0 | Prasad Konka | 3/9/2016 |  |  | Initial Draft |
| 1.0 | Zhengwu Lu | 3/16/16 |  |  | Review |
| 1.0 | Prasad Konka | 3/17/16 |  |  | Updates |
| 1.0 | Prasad Konka | 3/21/16 |  |  | Updates from George Zaki comments |
| 1.0 | Zhengwu Lu | 11/27/16 |  |  | Minor update |
| 1.0 | Prasad Konka | 2/23/2017 |  |  | API Security |
| 1.1 | Prasad Konka | 5/22/2016 |  |  | Service account, database setup |

TABLE OF CONTENTS

[1. Purpose 4](#_Toc483227375)

[2. Introduction 4](#_Toc483227376)

[2.1. What is HPC DME? 4](#_Toc483227377)

[2.2. Intended Users 4](#_Toc483227378)

[3. Setting up HPC DME API 5](#_Toc483227379)

[3.1. Pre-requisiteS 5](#_Toc483227380)

[3.1.1 Java 5](#_Toc483227381)

[3.1.2 Install Maven 5](#_Toc483227382)

[3.1.3 Install SVN Client 5](#_Toc483227383)

[3.1.4 Install ServiceMix 5](#_Toc483227384)

[3.1.5 Install PostgreSQL 5](#_Toc483227385)

[3.1.6 Install iRODS 5](#_Toc483227386)

[3.1.7 Install iRODS iCommands CLI 6](#_Toc483227387)

[3.2. Build and Deploy 7](#_Toc483227388)

[3.2.1 Set HPC DME Services properties 7](#_Toc483227389)

[3.2.2 Set up CA Certificate for SSL port 13](#_Toc483227390)

[3.2.3 Build HPC DME 14](#_Toc483227391)

[3.2.4 Build HPC DME keystore 14](#_Toc483227392)

[3.2.5 Deploy HPC DME into ServiceMix 14](#_Toc483227393)

[3.2.6 Setup HPC DME Database 15](#_Toc483227394)

[3.2.7 Import SSL Certs into Java Keystore 15](#_Toc483227395)

[4. Accessing HPC DME 15](#_Toc483227396)

[5. HPC DME Services ADMIN Operations 16](#_Toc483227397)

[5.1 Set up System Admin account 16](#_Toc483227398)

[5.2 Set up Service account 16](#_Toc483227399)

[5.2.1 Password Policy 18](#_Toc483227400)

[5.3 Set Up Base PATH for Each DOC 19](#_Toc483227401)

[5.4 Managing Security 19](#_Toc483227402)

[5.4.1 Roles 20](#_Toc483227403)

[5.4.2 Service API Security 21](#_Toc483227404)

[5.5 Create a User 23](#_Toc483227405)

[5.6 Updating an Existing User 24](#_Toc483227406)

[5.7 Assigning permissions to HPC DME User 24](#_Toc483227407)

# Purpose

The HPC DME Admin guide describes pre-requisites and installation instructions to setup HPC Data Management Services. The Admin guide further details different functions admin users would be performing to support HPC DME user needs.

# Introduction

## What is HPC DME?

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

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

HPC DME has two main components, HPC DME API – RESTful API and HPC DME Web – Web UI interacting with the API.

## Intended Users

The HPC DME has been built to cater to the data storage and data management needs of NCI cancer community. Any user with a valid HPC DME user account can run its client interface. This guide intends to guide HPC system administrators in managing user accounts, security and data integrity.

Note: HPC DME supports only NCI account holders at this point of time.

# Setting up HPC DME API

## Pre-requisiteS

### Java

Download and install Java 1.8.x

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

Please visit the following link for installation instructions

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

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

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

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

### Install Maven

Install Maven 3.3.3: [https://maven.apache.org/download.cgi#](https://maven.apache.org/download.cgi)

After successful installation of Maven, update PATH with Maven executable folder (Ex: c:/apache-maven-3.3.3/bin)

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

### Install SVN Client

Install Subversion client: <https://subversion.apache.org/packages.html>

### Install ServiceMix

Downlad Servicemix 7.0.0.M3: <http://servicemix.apache.org/downloads.html>

Extract ServiceMix to a folder.

### Install PostgreSQL

Install PostgreSQL 9.5.x for iRODS iCAT database

<https://help.ubuntu.com/community/PostgreSQL>

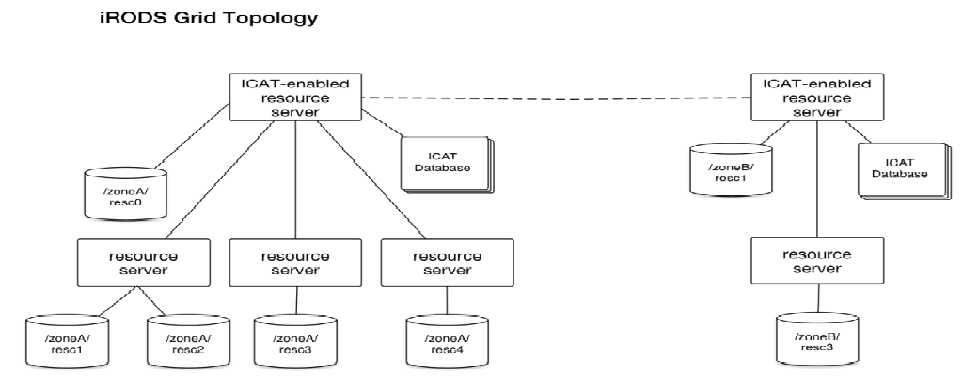
### Install iRODS

Download iRODS 4.2.x from the following link:

<http://irods.org/download/>

There are two types of iRODS server installation, iCAT and Resource Server:

1. An iCAT server manages a Zone, handles the database connection to the iCAT metadata catalog (which could be either local or remote), and can provide Storage Resources. An iRODS Zone will have exactly one iCAT server.
2. A Resource server connects to an existing Zone and can provide additional storage resource(s). An iRODS Zone can have zero or more resource servers.



An iCAT server is just a resource server that also provides the central point of coordination for the Zone and manages the metadata. A single computer cannot have both an iCAT server and a Resource server installed. The simplest iRODS installation consists of one iCAT server and zero Resource servers.

Please visit following link for detailed instructions on how to install iRODS iCAT and Resource servers. Current implementation of HPC DME supports setting up one iRODS zone with no resource servers: <https://docs.irods.org/master/manual/installation/>

### Install iRODS iCommands CLI

If you are planning to interact with iRODS remotely, download iCommands CLI package from <http://irods.org/download/>

The iCommands client is a command line interface that can be installed on Unix-based operating systems (Linux, Solaris, Macintosh, AIX) as well as Windows platforms.

This package installs just the client iCommands. Install this package if you want to have a system-wide installation of the iCommands (in /usr/bin), but do not want the entire iRODS iCAT or resource server. The iCommands are already included in both the iCAT and Resource packages – so installing this package in addition is unnecessary (and will conflict).

Once the iCommands client has been installed, you must then configure you iRODS environment. This can be done by editing $HOME/.irods/.irodsEnv. You can set the contents of this file using the following configuration.

{

"irods\_host": "<iRODS host name>",

"irods\_port": <iRODS port>,

"irods\_user\_name": "rods",

"irods\_zone\_name": "<Zone name>"

}

The default installation of iRODS comes with a system user account 'rods' with a default password 'irods'. You should change the password before letting anyone else into the system:

irods@hostname:~/ $ iadmin moduser rods password <newpassword>

To make sure everything succeeded, you will need to re-authenticate and check the new connection:

irods@hostname:~/ $ iinit

Enter your current iRODS password:

irods@hostname:~/ $ ils

/tempZone/home/rods:

## Build and Deploy

* Set your $PATH environment variable

PATH=$MAVEN\_HOME/bin:$SERVICE\_MIX\_HOME/bin:$PATH

* First, checkout code from HPC DME SVN repository:

Repository URL: <https://ncisvn.nci.nih.gov/svn/HPC_Data_Management>

Branch name: hpc-dm-release-1.1.0

* Set HPC\_HOME environment variable to the ‘src’ directory path in the source tree.

### Set HPC DME Services properties

HPC DME build scripts supports preparing the build for “dev”, “preprod” and “prod” environments.

Edit hpc-prototype-dev\src\hpc\hpc-server\hpc-features\src\main\resources\<properties file> to set system configuration properties.

To set a system property, you would need to uncomment it and set a valid value to it. If a property is commented, it uses its default value.

The following system properties are related to REST services

|  |  |  |
| --- | --- | --- |
| Property Name | Description | Default Value |
| **The following system properties are related to REST services** | | |
| hpc.ws.rs.host | This is host name of the machine where HPC DME Rest services will be running | localhost |
| hpc.ws.rs.port | This is the unsecured port of the machine where HPC DME Rest services will be running. You can turn off this port in ServiceMix | 7737 |
| hpc.ws.rs.secured-port | This is the secured port of the machine where HPC DME Rest services will be running. This port should be secured with CA cert | 7738 |
| hpc.ws.rs.path | HPC DME Server URL context name | hpc-server |
| hpc.ws.rs.stackTraceEnabled | Enable stacktrace as part of the error log | True |
| hpc.ws.rs.api-version | Do not change this value. This is updated by developers for any new release | 1.1.0 |
| hpc.ws.rs.ldapAuthentication | Enable NCI LDAP Authentication. If LDAP authentication is set to false, HPC DME uses authentication based on iRODS accounts | true |
| hpc.ws.rs.attachment-tmp-directory | Temporary staging directory where servicemix writes files while receiving from a client request |  |
| hpc.ws.rs.attachment-memory-threshold | Memory threshold before servicemix starts writing into “hpc.ws.rs.attachment-tmp-directory” folder | 1024 |
| hpc.ws.rs.ssl.keystore.path | Name of the keystore where HPC DME certificate is stored | ${karaf.base}/etc/hpc-server/keystore.jks |
| hpc.ws.rs.ssl.keystore.type | Type of the keystore where HPC DME certificate is stored | JKS |
| hpc.ws.rs.ssl.keystore.password | Password of the keystore where HPC DME certificate is stored | hpc-server-store-pwd |
| hpc.ws.rs.ssl.truststore.path | Name of the trust store where HPC DME certificate is stored | ${karaf.base}/etc/hpc-server/keystore.jks |
| hpc.ws.rs.ssl.truststore.type | Type of the trust store where HPC DME certificate is stored | JKS |
| hpc.ws.rs.ssl.truststore.password | Password of the trust store where HPC DME certificate is stored | hpc-server-store-pwd |
| hpc.ws.rs.ssl.wantClientAuth | Set this property to the value true to configure the server to request a client certificate. This property is ignored when configured for a client component such as SSLSocketAppender. | false |
| hpc.ws.rs.ssl.needClientAuth | Set this property to the value true to configure a server to require a valid client certificate. This property is ignored when configured for a client component such as SSLSocketAppender. | false |
| hpc.ws.rs.jetty-engine.minThreads | Specifies the min size of the thread pool used by the specific Jetty instance | 5 |
| hpc.ws.rs.jetty-engine.maxThreads | Specifies the max size of the thread pool used by the specific Jetty instance | 15 |
| **The following system properties are related to REST services access permissions** | | |
| hpc.ws.rs.auth.register-user | Mapping user registration function to Allowed roles to register user | SYSTEM\_ADMIN GROUP\_ADMIN NOT\_REGISTERED |
| hpc.ws.rs.auth.security.update-user | Mapping user registration function to Allowed roles to update user | SYSTEM\_ADMIN |
| hpc.ws.rs.auth.get-user | Allowed roles to query user | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.security.get-invoker | Allowed roles to query invoker | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.security.get-active-users | Allowed roles to query active users | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.security.get-all-users | Allowed roles to query all users | SYSTEM\_ADMIN |
| hpc.ws.rs.auth.authenticate | Allowed roles to authenticate user | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.security.register-group | Allowed roles to register a group | SYSTEM\_ADMIN GROUP\_ADMIN |
| hpc.ws.rs.auth.security.update-group | Allowed roles to update group | SYSTEM\_ADMIN GROUP\_ADMIN |
| hpc.ws.rs.auth.security.get-group | Allowed roles to get group | SYSTEM\_ADMIN GROUP\_ADMIN |
| hpc.ws.rs.auth.security.get-groups | Allowed roles to get groups | SYSTEM\_ADMIN GROUP\_ADMIN |
| hpc.ws.rs.auth.security.delete-group | Allowed roles to delete group | SYSTEM\_ADMIN GROUP\_ADMIN |
| hpc.ws.rs.auth.security.register-system-account | Allowed roles to register system account | SYSTEM\_ADMIN |
| hpc.ws.rs.auth.register-collection | Allowed roles to register collections | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.get-collection | Allowed roles to query a collection | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-management.download-collection | Allowed roles to download collection | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-management.set-collection-permissions | Allowed roles to set collection permissions | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-management.get-collection-permissions | Allowed roles to get collection permissions | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.register-data-object | Allowed roles to register a data object | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.get-data-object | Allowed roles to query data object | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-management.download-data-object | Allowed roles to download data object | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-management.set-data-object-permissions | Allowed roles to set data object permissions | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-management.get-data-object-permissions | Allowed roles to get data object permissions | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-management.get-data-management-model | Allowed roles to get data management model | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-management.get-data-management-tree | Allowed roles to get data management tree | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-search.query-collections | Allowed roles to query collections | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-search.query-data-objects | Allowed roles to query data objects | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-search.add-query | Allowed roles to save a named query | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-search.update-query | Allowed roles to update a named query | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-search.delete-query | Allowed roles to delete a named query | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-search.get-query | Allowed roles to get a named query | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-search.get-queries | Allowed roles to get a named queries | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-search.get-metadata-attributes | Allowed roles to get all metadata attributes with levels | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.data-search.refresh-metadata-views | Allowed roles to refresh search views in the database | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.notification.subscribe-notifications | Allowed roles to subscribe to system notifications | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.notification.get-notification-subscriptions | Allowed roles to get subscribe users to system notifications | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| hpc.ws.rs.auth.report.generate-report | Allowed roles to get generate reports | SYSTEM\_ADMIN GROUP\_ADMIN USER |
| **The following system properties are related to the application** | | |
| hpc.service.docBasePath | Base Path for each registered DOCs |  |
| hpc.service.security.authenticationTokenSignatureKey | Signature key for the authentication token |  |
| hpc.service.security.authenticationTokenExpirationPeriod | Authentication token expiration in mins | 120 |
| **The following system properties are related to business services called by REST API.** | | |
| hpc.bus.data-management.registration.endpoint | HPC DME Data archive system Globus Endpoint name | nihfnlcr#gridftp1 |
| hpc.bus.data-management.registration.path | HPC DME Data archive system Globus Endpoint base path | /mnt/IRODsTest/ |
|  |  |  |
| **The following system property is related to application services called by business services.** | | |
| hpc.service.user.DOC | Valid User division/center names (Abbreviations only). When registering a user, DOC value should be one of these values given here in the next cell. | FNLCR CCBR |
|  |  |  |
| **The following system properties are related to application services called by integration services.** | | |
| hpc.integration.irods.host | iRODS host name | fr-s-hpcdm-gp-d.ncifcrf.gov |
| hpc.integration.irods.port | iRODS host port number | 1247 |
| hpc.integration.irods.zone | iRODS zone name | tempZone |
| hpc.integration.irods.resource | iRODS resource name | demoResc |
| hpc.integration.irods.basePath | iRODS zone base path | / tempZone/home |
| hpc.integration.globus.nexusAPIURL | Globus API endpoint URL | nexus.api.globusonline.org |
| hpc.integration.globus.globusURL | Globus Online URL | www.globusonline.org |
| hpc.dao.postgresql.dbName | iRODS iCAT database name |  |
| hpc.dao.postgresql.host | iRODS iCAT database host name |  |
| hpc.dao.postgresql.port | iRODS iCAT database port number |  |
| hpc.dao.postgresql.username | iRODS iCAT database user name |  |
| hpc.dao.postgresql.password | iRODS iCAT database user password |  |
| hpc.dao.postgresql.encryptor.key | Encryptor key used to encrypt integrating system credentials | \*\*\*128-bit-key\*\* |
|  |  |  |

### Set up CA Certificate for SSL port

1. Locate Certificate files:
   1. Primary certificate – host.domain.crt
   2. Intermediate certification – DigiCertCA.crt
   3. Private Key – host.domain.key
2. Generate a Keystore & Import Intermediate Certification

*keytool –importcert –trustcacerts –file DigiCertCA.crt –alias root –keystore keystore.jks*  
You will be asked to create a password for the keystore (Make sure to remember it).

1. Combine Primary Certificate and Private Key

*openssl pkcs12 –export –name [domain] –in [host.domain.crt] –inkey [host.domain.key] –out keystore.p12*  
You will be asked to create a password. For simplicity, use the same as in step #2

1. Import the Primary Certificate & Private Key into the Keystore

*keytool –importkeystore –destkeystore keystore.jks –srckeystore keystore.p12 –srcstoretype pkcs12 –alias [domain]*

You will be asked to provide passwords. Use what you created in step #2.

You now have keystore.jks that contains the cert. Copy it to *$SERVICEMIX\_HOME/etc/hpc-server*

1. Configure HPC-DM

Uncomment and change these properties as needed to set the path and the password for the keystore.jks file.

$SERVICEMIX\_HOME/etc/gov.nih.nci.hpc.cfg:

#hpc.ws.rs.ssl.keystore.path=${karaf.base}/etc/hpc-server/keystore.jks

#hpc.ws.rs.ssl.keystore.password=hpc-server-store-pwd

#hpc.ws.rs.ssl.truststore.path=${karaf.base}/etc/hpc-server/keystore.jks

#hpc.ws.rs.ssl.truststore.password=hpc-server-store-pwd

### Build HPC DME

Open cmd window and navigate to $HPC\_HOME/

Run “*mvn clean install -P<env>*”

Valid values for <env> are dev, preprod, prod

### Build HPC DME keystore

To setup SSL port certificate with ServiceMix, you would create your own keystore with the certificate or could use build utility to create a keystore with self-signed certificate.

*cd $HPC\_HOME/hpc-server/hpc-ws-rs-impl*

*mvn clean install –Pkeystore*

### Deploy HPC DME into ServiceMix

Run Servicemix from the cmd prompt: servicemix

After successful start, type the following commands in the Servicemix Console

*features:addUrl mvn:gov.nih.nci.hpc/hpc-features/1.0.0-SNAPSHOT/xml/features*

*features:install hpc-server*

### Setup HPC DME Database

HPC DME users iRODS iCAT database to host its own schema. Once iCAT database is setup, please run following scripts to create HCP DME schema in iCAT Postgres database.

hpc-prototype-dev\src\hpc\hpc-server\hpc-dao-impl\src\main\scripts

If you are doing a new setup, please run all the scripts from “schema” folder. If you are already have HPC DME database schema setup, run scripts from migration folder based on your current version.

### Import SSL Certs into Java Keystore

HPC DME API integrates with Cleversafe, iRODS, NIH AD over SSL to enable secure communication.

* + 1. Please import NIH AD SSL Cert into Java keystore

keytool -importcert -file NIHDeviceRoot.pem -keystore cacerts -alias NIHRoot

keytool -importcert -file NIHDeviceIntermediate.pem -keystore cacerts -alias NIHIntermediate

* + 1. Please import Cleversafe SSL Cert into Java keystore

keytool -importcert -file Cleversafe.crt -keystore cacerts -alias Cleversafe

* + 1. Please import iRODS SSL Cert into Java keystore

keytool -importcert -file iRODS.pem -keystore cacerts -alias iRODS

# Accessing HPC DME

HPC DME can be accessed through its Service API or through its client interface. HPC DME Service APIs are developed based on REST standard. These APIs can be accessed by any interface that is compatible with REST standard. For example, these Service APIs can be accessed through Java, Perl, Ruby, Curl, JavaScript languages programmatically or through UI clients like SOAP UI, REST Client.

The following is the HPC DME Service API URL:

https://<host name>:<port>/<hpc.ws.rs.path>/<Resource Name>

HPC DME can also be accessed through its interactive client interface. Please refer to [HPC DME User guide](https://ncisvn.nci.nih.gov/svn/HPC_Data_Management/branches/hpc-prototype-dev/doc/guides/HPC_User_Guide.docx) and [API Specification](https://ncisvn.nci.nih.gov/svn/HPC_Data_Management/branches/hpc-prototype-dev/doc/guides/HPC_Server_API.pdf) for details.

# HPC DME Services ADMIN Operations

HPC Admin is responsible for HPC user accounts and authorizations setup to facilitate end users in using the solution to manage scientific data/metadata workflow. Managing hardware, storage, network is out of scope for this document.

## Set up System Admin account

HPC DME users are configured to get authenticated with NCI LDAP pool. When HPC DME is deployed initially, it does not have any accounts created. In order to set up a system admin account, please perform following steps one time only.

1. Login into PostgreSql database.
   1. psql <userId> -h <host> -d <database>
   2. After successfully login into the database, execute following statement

INSERT INTO "HPC\_USER"("USER\_ID", "FIRST\_NAME", "LAST\_NAME", , "IRODS\_USERNAME", "CREATED", "LAST\_UPDATED", , "IRODS\_PASSWORD", "DOC") VALUES ('<NCI Login ID>', '<First Name>', '<Last Name>', '<NCI Login ID>', '<MM-DD-YYYY>', '<MM-DD-YYYY>', null, 'FNLCR/CCBR');

1. Login into iRODS through its iCommands CLI and execute following command.

iadmin mkuser <NCI UserId>#<Zone name> Type rodsadmin

1. Assign permission on iRODS home to System Admin

ichmod -r own <adminUserId> <iRODS home>

## Set up Service account

HPC DME uses service account to integrate with Globus, Cleversafe and iRODS. While HPC DME security enforces authentication and authorization, the service account is used to perform actions on behalf of the users. For example, the service account is used to upload data into Cleversafe data archive object store. It is used to update system metadata in iRODS. It is also used to manage asynchronous data transfer using Globus. It is important to keep service accout credentials in a safe location. HPC DME stores service accounts (Globus, Cleversafe and iRODS) information in its database in an encrpted manner. HPC DME provides a REST API to register service account with the system initially. This API is accessible to System Administrator role only. Please see API Specification section below for API details.

The following service account credentials are expected to be setup and attached with the service account for HPC DME to function properly.

1. Cleversafe AWS access key and secret access key

2. iRODS user name and password with “rodsadmin” role and has “own” access to the root folder

3. Globus user name and password with read/write access to “nihfnlcr#gridftp1”

**To setup Cleversafe credentials using curl:**

|  |
| --- |
| Set Cleversafe credentials into a file called Cleversafe.json  {  "account": {  "username": "AWS Secret key",  "password": "AWS Secret access key",  "integratedSystem": "CLEVERSAFE"  },  "dataTransferType" : "S3"  }  Execute following curl command with correct parameter values:  curl -H "Content-Type: application/json" -d @Cleversafe.json -X PUT <server URL>/user --user <userId>:<password> -H "Accept: application/json" |

**To setup Globus credentials using curl:**

|  |
| --- |
| Set Globus credentials into a file called Globus.json  {  "account": {  "username": "Globus UserId",  "password": "Globus Password",  "integratedSystem": "GLOBUS"  },  "dataTransferType" : "GLOBUS"  }  Execute following curl command with correct parameter values:  curl -H "Content-Type: application/json" -d @Globus.json -X PUT <server URL>/user --user <userId>:<password> -H "Accept: application/json" |

**To setup iRODS credentials using curl:**

|  |
| --- |
| Set iRODS credentials into a file called iRODS.json  {  "account": {  "username": "iRODS UserId",  "password": "iRODS Password",  "integratedSystem": "IRODS"  }  Execute following curl command with correct parameter values:  curl -H "Content-Type: application/json" -d @iRODS.json -X PUT <server URL>/user --user <userId>:<password> -H "Accept: application/json" |

### Password Policy

Globus service account is created with Globus.org and is linked with NCI AD Service account. Whenever there is an update to Globus service account credentials, they need to be updated in HPC DME database. Please use HPC DME API to update Globus credentials in HPC DME database.

CleverSafe credentials are generated by ITOG Group and they are in the format of S3 keys. Again, whenever there is an update to these keys, they need to be updated in HPC DME database. Please use HPC DME API to update Globus credentials in HPC DME database.

iRODS is setup to use PAM authentication with NIH AD. So iRODS uses NIH AD service account credentials. Following NIH password policy, this account credentials must be updated. whenever there is an update to these credentials, please use HPC DME API to update iRODS credentials in HPC DME database.

## Set Up Base PATH for Each DOC

Each DOC in HPC DME is assigned with a base path. This base path will have to be created in the iRODS system by the System Admin.

|  |
| --- |
| Edit and save the following JSON into basepath.json file.  {  "metadataEntries": [  {  "attribute": "name",  "value": "Base path for DOC XYZ"  }  ]  }  Execute following curl command with correct parameter values:  curl -H “Content-Type: application/json” -d @basepath.json -X PUT <server>/collection/<Base path Name> -H "Accept: application/json" -D <response-header> -o <response-message.json |

## Managing Security

HPC DME has different layers of security. First layer of security is enforced at HPC DME API level through NCI LDAP credentials. By default, HPC DME is setup to work with NCI LDAP pool server “ldaps://ncids4a.nci.nih.gov:636”. Creating or updating NCI LDAP user credentials is out of scope for HPC DME API.

HPC DME integrates with iRODS for data management and with Globus for data transfer. iRODS instance supports PAM authentication that integrates its authentication with NCI LDAP server. Please see the following link to setup PAM authentication for iRODS.

<https://github.com/DICE-UNC/jargon/wiki/Setting-up-PAM>

Execute following command to verify PAM authentication setup

/var/lib/irods/iRODS/server/bin/PamAuthCheck

In order to authenticate iRODS user with NCI LDAP, iRODS users should be created with NCI loginId only, password is not required.

HPC DME stores iRODS, Globus and Cleversafe service account credentials in its PostgreSql database in encrypted format. HPC DME API uses AES encryption in combination of encryption key set in HPC DME services properties. The following is a property in gov.nih.nci.hpc.cfg. Access to this file should be restricted.

hpc.dao.postgresql.encryptor.key=\*\*\*128-bit-key\*\*

### Roles

HPC DME supports the following roles:

1. USER – HPC DME user role with access to the following functions:
   * Update self-profile
   * Register a data object with metadata
   * Update metadata on data object
   * Download data object
   * Share data object
   * Search collections and data objects
   * Generate data object reports
2. GROUP\_ADMIN – HPC DME group admin role with access to following functions in addition to all USER role allowed functions:
   * List group info
   * Create a group
   * Create user
   * Add user to a group
   * Remove user from a group
   * Register a collection with metadata
   * Update collection metadata
   * Remove Collection
   * Remove Data Object
   * Generate collection reports
3. SYSYEM\_ADMIN - HPC DME system admin role with access to following functions in addition to all GROUP\_ADMIN role allowed functions:
   * Update User
   * Remove User
   * Query User Info
   * Generate system usage reports

HPC DME security is integrated with iRODS security. iRODS roles “rodsadmin”, “groupadmin”, “rodsusers” are mapped to HPC DME types as follows:

SYSTEM\_ADMIN 🡪 rodsadmin

GROUP\_ADMIN 🡪 groupadmin

USER 🡪 rodsusers

### Service API Security

Access to each of HPC DME API services is configured through Spring Security configuration at \hpc-server\hpc-ws-rs-impl\src\main\resources\META-INF\spring\ hpc-ws-rs-bundle-context-osgi.xml. Making a change to this file requires rebuilding the API packages and deploying it to run-time environment to have an effect. This configuration is not expected to change without approval from the Project Management.

|  |
| --- |
| <prop key="hpc.ws.rs.auth.security.register-user">SYSTEM\_ADMIN</prop>  <prop key="hpc.ws.rs.auth.security.update-user">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.security.get-user">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.security.set-group">SYSTEM\_ADMIN GROUP\_ADMIN</prop>  <prop key="hpc.ws.rs.auth.security.authenticate-user">SYSTEM\_ADMIN</prop>  <prop key="hpc.ws.rs.auth.security.register-system-account">SYSTEM\_ADMIN</prop>    <prop key="hpc.ws.rs.auth.data-management.register-collection">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-management.get-collection">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-management.register-data-object">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-management.get-data-object">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-management.download-data-object">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-management.set-permissions">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-management.get-data-management-model">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>    <prop key="hpc.ws.rs.auth.data-search.query-collections">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-search.query-data-objects">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-search.add-query">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-search.update-query">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-search.delete-query">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-search.get-query">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-search.get-queries">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-search.get-metadata-attributes">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.data-search.refresh-metadata-views">SYSTEM\_ADMIN</prop>    <prop key="hpc.ws.rs.auth.notification.subscribe-notifications">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>  <prop key="hpc.ws.rs.auth.notification.get-notification-subscriptions">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop>    <prop key="hpc.ws.rs.auth.report.generate-report">SYSTEM\_ADMIN GROUP\_ADMIN USER</prop> |

## Create a User

System Admin is responsible for creating HPC user accounts. System Admin can do that by accessing HPC DME REST API. Any REST client can be used to access the API.

For example, a system admin would perform the following using curl.

* Create input.json with the following content

|  |
| --- |
| {  "firstName": "John",  "lastName": "Doe",  "doc" : "FNLCR",  "userRole" : "SYSTEM\_ADMIN"  } |

Execute the following command:

*curl -H "Content-Type: application/json" -d @input.json -X PUT $<SERVER>/user/<UserId> --user <Admin NCIUserId:Password> -H "Accept: application/json"*

Valid values for “dataManagementUserType” are "SYSTEM\_ADMIN", "GROUP\_ADMIN" and "USER". Known issue: If you create a user with "GROUP\_ADMIN" role, iRODS Jargon API has a known issue in creating that. A workaround is to create user with “USER” role and update User role with “groupadmin” through iCommand interface. Once you create the user, you would need to login into iRODS iCommand and execute following command to assign “groupadmin” role. This is a workaround until the issue with iRODS Jargon API is fixed.

*iadmin moduser <userId>#<ZoneName> Type groupadmin*

Please see User registration API specification for more details on the API and any errors.

## Updating an Existing User

System Admin is responsible for updating HPC user accounts. System Admin can do that by accessing HPC DME REST API. Any REST client can be used to access the API.

For example, a system admin would do the following using curl.

* Create input.json with the following content

|  |
| --- |
| {  "firstName": "Eran",  "lastName": "Rosenberg",  "doc" : "FNLCR",  "userRole" : "SYSTEM\_ADMIN"  } |

Execute the following command:

*curl -H "Content-Type: application/json" -d @input.json -X POST $<SERVER>/user/<UserId> --user <Admin NCIUserId:Password> -H "Accept: application/json"*

Please see User registration update API specification for more details on the API and any errors.

## Assigning permissions to HPC DME User

In order to complete user registration with HPC DME and start using HPC DME API, a system administrator should complete the following steps for each user created.

* User should be NCI user with account in NCI Novell system
* Assign “own/write/read” permission to archive folder to the user.

|  |
| --- |
| Edit ad save the following json into permission.json  {  "entityPermissionRequests": [  {  "path": "/<DOC Base Path> ",  "userPermissions": [  {  "userId": "<UserId>",  "permission": "WRITE"  }  ]  }  ]  }  curl -H “Content-Type: application/json” -d @permission.json -X POST <server>/acl -H “Accept: application/json” -D <response-header> -o <response-message.json> |