# **High Performance Computing (HPC) Data Management Environment (DME) Getting Started: DOC Group Administrators**

## Purpose

This document is intended to serve as a guide for getting started on using the High Performance Computing (HPC) Data Management Environment (DME) for the National Cancer Institute (NCI). Specifically, this guide is intended to help the reader understand the duties of a DOC Group Administrator, should the reader be designated such role in facilitating a DOC’s utilization of the HPC DME.

## General Duties

A DOC Group Administrator (Group Admin) is expected to be a knowledgeable user of the HPC DME, serving as a technical leader and guide in his/her respective DOC. Therefore, he/she plays a critical role in the adoption and ongoing utilization of the HPC DME by others who belong to the same DOC.

**Setting up Metadata Policy**

One of the responsibilities of a Group Admin is creation and/or maintenance of the metadata policies for their DOC. The metadata policies are rules and conventions that must be followed by HPC DME users to put or update content in the DME. Within the DME, one DOC may be designated one or more base paths, and for each base path, there may be one set of policies set in effect. The set is divided into 3 parts. The first is the **data hierarchy specification**. It defines the permitted structure and organization of Collections under the applicable base path. The second part is the **Collections metadata validation rules**. It contains definitions of the expected metadata accompanying Collections and sets rules such as which metadata attributes are mandatory and what are acceptable values for a metadata attribute. The third part is the **Data Objects metadata validation rules**.

It is strongly recommended that every DOC utilizing the DME has comprehensive metadata policies sets for all its base paths in the DME. With well-devised metadata policies sets in effect, the DME users of a DOC shall find that Data Objects registered in the DME shall be easier to index, search, and organize, and treatment of Data Objects can be more consistent and uniform as they all conform to the same rules concerning metadata. This is critical to promote data sharing among different DOCs and across organizations.

Currently, the mechanism for creating or updating metadata policies sets is a manual procedure that an HPC DME System Administrator (Sys Admin) executes. Each metadata policies set is composed as 3 JavaScript Object Notation (JSON) documents for the 3 parts of a policies set: **data hierarchy specification**, **Collections metadata validation rules**, and **Data Objects metadata validation rules**. If a DOC Group Admin is comfortable working with JSON, he/she could, without the aid of a Sys Admin, write or edit these JSON documents. Any text editor application could be used for this purpose. Refer to the figures in the next section for examples of the 3 parts of a metadata policies set expressed in the expected JSON form and applicable to one base path.

The Sys Admin could assist with the mechanics of creating or updating a policies set, but ultimately, a Group Admin is responsible for producing the content in his/her DOC’s metadata policies set.

**Administering Users and Groups**

Another responsibility of an HPC DME Group Administrator is administering users and groups for his/her DOC. Tasks related to this responsibility can be accomplished via the HPC DME web GUI client. For details about that, please refer to the [HPC DATA MANAGEMENT USER GUIDE](https://github.com/CBIIT/HPC_DME_APIs/raw/master/doc/guides/HPC_User_Guide.docx).

## Example JSON Documents

Figure 1: Example of Data Hierarchy Specification

|  |
| --- |
| {  "collectionType": "PI\_Lab",  "isDataObjectContainer": false,  "subCollections": [  {  "collectionType": "Project",  "isDataObjectContainer": true,  "subCollections": [  {  "collectionType": "Flowcell",  "isDataObjectContainer": true,  "subCollections": [  {  "collectionType": "Sample",  "isDataObjectContainer": true  }  ]  }  ]  }  ]  } |

The following is a summary of the attributes in a JSON document defining a Data Hierarchy Specification. Note that the Data Hierarchy is effectively a tree of Collections.

1. **collectionType**: The kind of Collection permitted at position in Data Hierarchy
2. **isDataObjectContainer**: Flag to indicate if it the Collection may contain Data Objects; valid values are “true” and “false”
3. **subCollections**: Specifies Collections that a given Collection is allowed to contain

Figure 2: Example of Collections Metadata Validation Rules

|  |
| --- |
| {  "metadataValidationRules": [  {  "attribute": "collection\_type",  "mandatory": true,  "validValues": [  "Project",  "PI\_Lab",  "Flowcell",  "Sample",  "Folder"  ],  "ruleEnabled": true  },  {  "attribute": "pi\_name",  "mandatory": true,  "collectionTypes": [  "PI\_Lab"  ],  "ruleEnabled": true  },  {  "attribute": "project\_start\_date",  "mandatory": false,  "defaultValue": "System-Date",  "collectionTypes": [  "Project"  ],  "ruleEnabled": true  }  ]  } |

Figure 3: Example of Data Objects metadata validation rules

|  |
| --- |
| {  "metadataValidationRules": [  {  "attribute": "object\_name",  "mandatory": true,  "collectionTypes": [  "Sample"  ],  "ruleEnabled": true  },  {  "attribute": "phi\_content",  "mandatory": true,  "collectionTypes": [  "Sample"  ],  "defaultValue": "Unspecified",  "validValues": [  "Unspecified",  "PHI Present",  "PHI Not Present",  "Not Specified"  ],  "ruleEnabled": true  },  {  "attribute": "data\_compression\_status",  "mandatory": true,  "collectionTypes": [  "Sample"  ],  "defaultValue": "Unspecified",  "validValues": [  "Unspecified",  "Compressed",  "Not Compressed",  "Not Specified"  ],  "ruleEnabled": true  }  ]  } |

The following is a summary of the attributes in each JSON object defining a metadata validation rule element regardless of whether the context is Collections or Data Objects.

1. **attribute**: Metadata attribute name
2. **mandatory**: Flag to indicate if it is required or not. Valid values are “true”, “false”
3. **defaultValue**: Default value of the attribute if no value is given
4. **collectionType**: Collection type name applicable for this attribute rule. Only one value is allowed.
5. **validValues**: List of valid values for this attribute
6. **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.
7. **DOC**: Division name applicable. Any registered user account is assigned one DOC in the system. Any specific DOC dependent rules may be attached and activated.