*HPC Data management*

System Integration Developer Guide

Version *1.0*

*5/17/2017*

**Version History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version Number** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Description of Change** |
| 1.0 | *Eran Rosenberg* | *5/17/2017* |  |  | *Initial Draft* |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

TABLE OF CONTENTS

1 Introduction 4

1.1 Purpose of This Document 4

1.2 Integrated Systems 4

1.3 HPC DM API Server Architecture 5

1.4 Code Organization 6

2 iRODS Integration 7

2.1 Jargon API Integration Points 7

2.2 Materialized Views Integration Points 8

3 Globus Integration 10

4 Cleversafe Integration 11

# Introduction

## Purpose of This Document

The HPC DM API server provides a set of data management REST services to deposit, annotate, search and retrieve large data objects. The API is implemented by combining capabilities of several external systems. This document provides the technical details on the how the external systems integration was implemented and lists the resulted dependencies.

This document is targeted for developers maintaining the HPC DM API server. It’s recommended to be reviewed before a planned upgrade or replacement of an integrated system.

## Integrated Systems

The HPC DM API Server is integrated with the following external Systems:

**iRODS**   
An open source data management solution. iRODS provides data registration with metadata catalog. It provides data search, security, audit, rule-engine and data transfer capabilities. (<https://irods.org>).

The integration with iRODS was developed using the Jargon API (<https://github.com/DICE-UNC/jargon/wiki/Jargon-overview>). There are several requirements that could not be satisfied with Jargon, and the server is querying the iRODS DB directly.

**Globus**   
An open source data transfer and sharing platform. Globus provides ability to transfer large files asynchronously, securely and reliably. (<https://www.globus.org/>)

The integration with Globus was developed using the Globus transfer API using Globus nexus API to authenticate w/ Globus. Both are Java jars from Globus that are packaged with the API server. The transfer API provides convenient Java API to communicate with the Globus REST services (https://docs.globus.org/api/transfer).

**Cleversafe**

IBM’s object storage solution, which is deployed on-premises but can be deployed to public or dedicated cloud. Note that Cleversafe supports the Amazon S3 data transfer API. (<https://www.ibm.com/cloud-computing/products/storage/object-storage/>).

The integration with Cleversafe was developed using the Amazon AWS SDK for Java (<https://aws.amazon.com/sdk-for-java>).

**Active Directory**

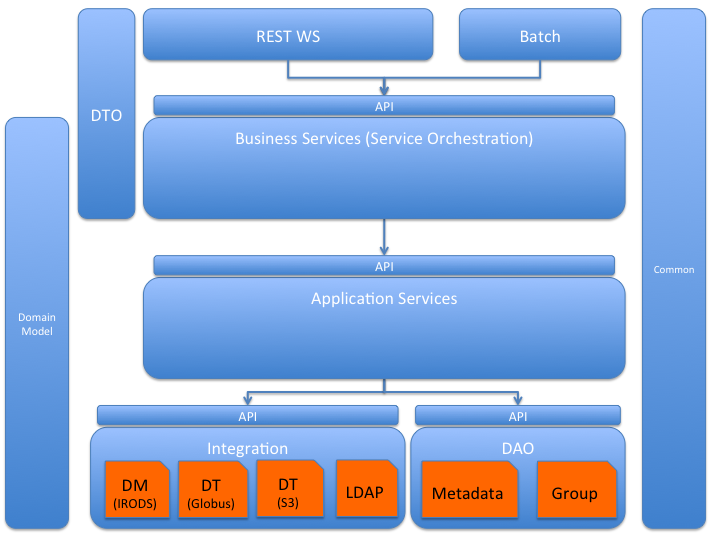
Microsoft’s directory and authentication system. The integration was done using the Java standard naming API, and is trivial so not covered in this document.

## HPC DM API Server Architecture

The following diagram depicts the HPC DM API Server architecture components and their dependencies. The code that integrates with the external systems is co-located in the ‘Integration’ layer. Like any other layer in the architecture, the integration layer provides a pure Java API to use it, and any integrated-system specifics are abstracted or hidden from the caller.

The ‘Application Services’ (within the app services layer) are the only components that use the integration code. These application services use the integration services as building blocks to implement their functionality.

There are several



## Code Organization

The integration code can be found in the following maven projects and is organized in the following java packages:

* **hpc-integration-api**
  + *gov.nih.nci.hpc.integration* – package contains all integration interfaces (This is the pure Java API the ‘application services’ are using to invoke the integration services)
* **hpc-integration-impl**
  + *gov.nih.nci.hpc.integration.globus.impl* – Globus implementation of the Data Transfer Integration API
  + *gov.nih.nci.hpc.integration.s3.impl* – S3 implementation of the Data Transfer Integration API .
  + *gov.nih.nci.hpc.integration.irods.impl* – Implementation of the Data Transfer Management Integration API
  + *gov.nih.nci.hpc.integration.ldap.impl* – LDAP implementation of the Security Authentication Integration API .

The iRODS integration includes direct querying of iRODS DB. The relevant source code can be found in the following maven project

* **hpc-dao-impl**
  + *hpc\_hierarchical\_metadata.sql* – This is a script to create a set of materialized views based on iRODS table. HPC DM provides a capability to search data object by metadata defined anywhere in the object ‘hierarchy’ (i.e. metadata that are associated with the containing collection hierarchy tree). The purpose of the materialized view is to generate that ‘hierarchical metadata’ table, so search queries can be easily implemented.
  + *HpcMetadataDAOImpl.java* – Implements the data search by querying the hierarchical metadata materialized view.
  + *HpcGroupDAOImpl* – HPC DM provides a capability to search for user groups using case insensitive matching. This capability is not available via the Jargon API, so the implementation queries the iRODS table directly.

# iRODS Integration

## Jargon API Integration Points

Jargon is a Java API provides a convenient way to interact with iRODS. The current Jargon version used is 4.1.10.0. The following table lists the integration points that are implemented with Jargon and the specific API that was used:

| **Integration Point** | **Jargon API** |
| --- | --- |
| authenticate | IRODSAccessObjectFactory.*authenticateIRODSAccount*()  AuthResponse.*getAuthenticatedIRODSAccount*() |
| disconnect | IRODSAccessObjectFactory.*closeSessionAndEatExceptions*() |
| createCollectionDirectory | IRODSFileFactory.*instanceIRODSFile*()  IRODSFile.*mkdirs*() |
| createDataObjectFile | IRODSFileFactory.*instanceIRODSFile*()  IRODSFile.*createNewFile*() |
| delete | IRODSFileFactory.*instanceIRODSFile*()  IRODSFile.deleteWithForceOption() |
| addMetadataToCollection | CollectionAO.*addBulkAVUMetadataToCollection()* |
| updatCollectionMetadata | CollectionAO.modifyAvuValueBasedOnGivenAttributeAndUnit*()* |
| addMetadataToDataObject | DataObjectAO.*addBulkAVUMetadataToDataObject()* |
| updatDataObjectMetadata | DataObjectAO.modifyAvuValueBasedOnGivenAttributeAndUnit*()* |
| getPathAttributes | IRODSFileFactory.*instanceIRODSFile*() |
| getCollection | CollectionAO.*findByAbsolutePath()*  CollectionAndDataObjectListAndSearchAO.*list DataObjectsAndCollectionsUnderPath()* |
| getCollectionMetadata | CollectionAO.findMetadataValuesForCollection*()* |
| getDataObject | DataObjectAO.*findByAbsolutePath()* |
| getDataObjects | DataObjectAO.*findDomainByMetadataQuery()* |
| getCollectionMetadata | DataObjectAO.findMetadataValuesForDataObject*()* |
| addUser | UserAO.*addUser()* |
| deleteUser | UserAO.*deleteUser()* |
| updateUser | UserAO.*updateUser()* |
| getUserRole  userExists | UserAO.*findByName()* |
| getCollectionPermissions | CollectionAO.listPermissionsForCollection*()* |
| getCollectionPermissionForUser | CollectionAO.getPermissionForUserName*()* |
| setCollectionPermission | CollectionAO.setAccessPermission*()* |
| getDataObjectPermissions | DataObjectAO.listPermissionsForDataObject*()* |
| getDataObjectPermissionForUser | DataObjectAO.getPermissionForDataObjectForUserName*()* |
| setDataObjectPermission | DataObjectAO.setAccessPermission*()* |
| addGroup | UserGroupAO.*addUserGroup()* |
| deleteGroup | UserGroupAO.*removeUserGroup()* |
| groupExists | UserGroupAO.*findWhere()* |
| addGroupMember | UserGroupAO.*addUserToGroup()* |
| deleteGroupMember | UserGroupAO.*removeUserFromGroup()* |
| getGroupMembers | UserGroupAO.*listUserGroupMembers()* |

## Materialized Views Integration Points

To support hierarchical metadata for collections and data objects, 6 materialized views were created. The following tables depict the dependencies of the materialized views on iRODS DB tables:

|  |  |
| --- | --- |
| **HPC DM Materialized View** | **Dependent iRODS tables** |
| r\_coll\_hierarchy\_matamap | r\_coll\_main  r\_objt\_metamap |
| r\_coll\_hierarchy\_metamain | r\_coll\_hierarchy\_metamap  r\_meta\_main  r\_objt\_metamap |
| r\_coll\_hierarchy\_meta\_attr\_name | r\_coll\_hierarchy\_metamap |
| r\_data\_hierarchy\_matamap | r\_data\_main  r\_objt\_metamap |
| r\_data\_hierarchy\_metamain | r\_data\_hierarchy\_metamap  r\_meta\_main  r\_objt\_metamap |
| r\_data\_hierarchy\_meta\_attr\_name | r\_data\_hierarchy\_metamap |

The following table lists the integration points that are implemented with the materialized views / iRODS DB

|  |  |
| --- | --- |
| **Integration Point** | **Materialized View / iRODS DB table** |
| getCollectionPaths  getCollectionCount | r\_coll\_hierarchy\_meta\_main  r\_objt\_access  r\_user\_main  r\_user\_group |
| getDataObjectPaths  getDataObjectCount | r\_data\_hierarchy\_meta\_main  r\_objt\_access  r\_user\_main  r\_user\_group |
| getCollectionMetadata | r\_coll\_hierarchy\_meta\_main |
| getDataObjectMetadata | r\_data\_hierarchy\_meta\_main |
| getCollectionMetadataAttributes | r\_coll\_hierarchy\_meta\_attr\_name  r\_objt\_access  r\_user\_main  r\_user\_group |
| getDataObjectMetadataAttributes | r\_data\_hierarchy\_meta\_attr\_name  r\_objt\_access  r\_user\_main  r\_user\_group |
| getGroups | r\_user\_main |

# Globus Integration

# Cleversafe Integration