**iRODS GENI plan**

We present a summary of the plan first and leave the detailed discussions inline.

# Summary of effort

To achieve the goals described in Section II, the following tasks need to be done.

1. Develop a micro-service that processes a specified metadata file (e.g. bagit.txt) and stores the metadata in iCAT.

2. Collaborate with CNRI to decide the integration of iRODS and Handle System. Depending on the approach we agree upon, it may require developing a micro-service to register/obtain a handle from the Handle System. In addition, some kind of caching mechanism may need to be developed to cache the handle to iCAT mapping for users.

3. May need to modify the GUI (web browser and/or iDrop) to show and query the Handle.

4. Define rules (policies) for archiving data from the User workspace to long-term archive.

5. Once the archiving rules are defined, we can create an iCommand (e.g. iarchive) to simplify the operations.

6. iRODS supports GSI (Grid Security Infrastructure) which is certificate based and uses SSL. Version 3.2 of iRODS will support Pluggable Authentication Modules. Goals include InCommon authentication. We need to understand whether this is sufficient or not. For example, if ABAC is required, we need to study how it can be supported in iRODS.

# Discussions

The following goals are excerpted directly from Harry’s document.

 1. ***First goal:  A structured place to store all of the objects for an experiment, with descriptors (metadata), that is easy to access, with short to medium term storage, and the ability to search.  (Note:  this goes well beyond just measurement data objects.)***

There are two ways to implement this in iRODS. First is to use iCAT to store the descriptors. In iRODS, metadata is in AVU format: Attribute, Value and optionally Unit. Users can add metadata using iCommands such as ‘imeta’. The benefit of using iCAT to store the descriptors is it provides powerful and flexible SQL based query capability. Second is to use a metadata file. For example, using BagIt, a bag looks like:

mybag/

|--bagit.txt

|--manifest.txt

|--data/

bagit.txt can be used to store the descriptors. Currently, iRODS has an iCommand called ‘ilocate’, which is similar to ‘find’ in Unix. ‘ilocate’ supports search patterns that can be used to locate the metadata file in irods (e.g. mybag/bagit.txt). Then grep can be used to parse the file or users can develop their own parsers depending on the format of the metadata, which is not specified in iRODS.

Both approaches have pros and cons, and what we plan to do is a hybrid mode – we develop a micro-service for iRODS so that it will take care of the metadata file automatically. Specifically, it parses the metadata file and insert descriptors into iCAT. If the metadata is modified, the modifications will be automatically reflected in iCAT. Currently, iCAT needs to know every modification in the data it manages anyway, so it wont require a lot of work to achieve what we want. This approach gives the users two options to search: SQL based database searching and text based searching. Another advantage of this approach is that there are some common metadata such as when the data is created, how it is created, etc. are already maintained by the current iCAT; in the metadata file we can leave these common metadata to iRODS to handle and only focus on special ones.

 ***2. Second goal:  A separate long-term archive, with controlled access from the outside world, using a DOI (handle) as a persistent identifier***

A. A simple implementation is layering the Handle System on top of iRODS. As shown in the figure below, the User queries the Handle System to figure out who (a local handle service) can resolve the handle. Conceptually, the key component of such an implementation is the mapping from the handle (in the format of prefix/suffix) to a logical name in iRODS (e.g., /zone\_name/path/2/data). There are different ways to implement the mapping. For example, there may exist a one-to-one mapping between the zone name (as well as the iCAT’s IP) and the prefix. The mapping is saved in the Handle System and is used to direct a query to the corresponding iCAT. This implementation of mappings requires each zone to obtain a prefix (by registering with the Handle System). Alternatively, there may exist a Local Handle Service that delegates all the queries to GENI zones (shown as the dotted box in the figure below). This approach is probably more appropriate because the LHS can be managed and administrated by GENI.

The above-described approaches may not be efficient because each query needs to be resolved at least twice (one by the Handle system to find the corresponding iCAT, then the iCAT needs to resolve the query to the physical location). Because the mapping between the prefix to zone name is relatively stable, caching can be implemented for Users so that the first step of query (from handle to zone name by Handle System) can be skipped if the mapping can be found in the local cache.

 B. Another implementation is simply embed the handle into iRODS. For example, suppose each zone has a prefix obtained from the Handle System, the handle zone\_prefix/suffix can be viewed as an attribute associated with the logical name zone/path/2/data and saved in iCAT as a key. This way, the user command (e.g. iget) can use either the logical name or the handle of the data. Note that this approach requires exchanging the zone\_prefix when zones are federated so that if the user queries a data belonging to a different zone that she belongs to, iCAT needs to know which iCAT is responsible for the resolution from the handle.

Approaches A and B are fundamentally different in the sense that A has a centralized (conceptually) Handle System and B is completed distributed (therefore it relies on federation of data zones). B uses the Handle System only to obtain the handles – queries are handled directly by iCATs. Therefore it is more efficient in name resolution (one-step vs. two-step).

 ***3. Third goal:  Include most of the functionality provided Measurement Data Archive (MDA) prototype, built by CNRI.***

According to the MDA prototype document prepared by CNRI, the functionality provided includes

 a. separation of user workspace from long-term archive

 b. user workspace is Linux based allowing SMB/SFTP/SSH access

 c. user workspace can be searched and queried (using handles) using repository tools and processes

 d. archive is implemented by another instance of Digital Object Repository with its own storage

 e. web interface to workspaces that can search data using keywords or timestamps

 f. API support for SMB/HTTP

If necessary, iRODS can mimic all features the CNRI prototype provides. In iRODS, data’s life cycle can be separated into multiple stages by applying policies. Policies (implemented by rules) are flexible and powerful enough to achieve the separation of user workspace and archive. For example, we can create a logical name space (could be part of each zone or on a separate zone dedicated for archives) on reliable storage systems so that users can simply archive data from their workspace using ‘icp’. Some policies can be enabled such that ‘icp’ can automatically replicate data to separate locations and compute checksums so as to enhance the availability. In addition, rules can be scheduled periodically to check the integrity of data (e.g. ichecksum) and if something wrong, more replicas will be created. Again, the archives can be associated with the Handle System as discussed in Section 2.

iRODS supports SMB/SFTP/SSH and provides Web based, GUI (iDrop) and command line (iCommands) user interfaces . It also provides Java API (Jargon). All provide searching capability to satisfy different level of requirements. Since the Handle System provides Java API, it will be straightforward to talk to it. The iRODS Java RESTful interface is in the evaluation stage at this point.

 ***4. Establish multiple federated iRODS services, starting at***

***RENCI and UMass Amherst, and operate for GENI users (experimenters)  (GEC14)***

Currently there is only one iRODS zone (defined as geniRenci) in which the iCAT server is located at RENCI and data servers are located at both RENCI and UMass. This zone is only for the GIMI project, therefore it is not necessary to be federated. We can definitely create multiple federated zones if necessary. For example, if ExoGeni stores its templates (to create slices) in an iRODS zone (say, exogeni/), we can federate with exogeni so that GIMI can use the templates to create its own slices.

 ***5. Establish persistent accounts for each user, use icommands***

***to store and retrieve measurement data objects for each user in a storage service   (GEC14)***

Done.

 ***6. Establish authentication for each user based on certificates, and also proxy (delegated) certificates.  (GEC15)***

We need to know more about what kind of certificates and access control should be used. iRODS uses a secure password system, and also can be configured to support GSI and/or Kerberos. In iRODS, the access control/authorization info is managed by iCAT in a centralized fashion. Therefore the authentication part is separate from the authorization. iRODS supports GSI (Grid Security Infrastructure) which is certificate based and uses SSL. iRODS 3.2 will support Pluggable Authentication Modules. Goals include InCommon authentication.

 ***7. Establish directory structure in storage service for each user to accommodate multiple experiments, and directory structure for each experiment (consider "bag") to include all objects (artifacts) associated with that experiment, including one or more descriptors (metadata) within XML files (following the GENI descriptor schema)  (GEC15)***

There are indeed two approaches to this goal. First, a little bit of explanation of iRODS.

iRODS separates the logical name space from the physical name space. The logical name is hierarchical (e.g. /gimi\_zone/shuang/iperf\_experiment/data) as well as the physical name (geni\_gimi.renci.org:/home/shuang/iperf/data). Therefore, the directory structure can be organized in both the logical name space and physical name space.

The first approach is simple. For each user/experiment, a physical directory can be created and collect related data into sub-directories in whatever way the user wants. This requires moving data around and making copies if some data belong to multiple users/experiments. Another approach that iRODS provides is to create directories in the logical name space – e.g., /gimi\_zone/shuang/iperf\_experiment represents a virtual collection of data belong to this experiment. The physical data are possibly distributed and stored in different physical locations (and directories) but they form a logical “bag”, which includes “pointers” to the actually physical data. This approach is more flexible comparing with the former one because data are not moved/replicated in order to be shared between different users/experiments. Consider the case that an experiment requires to create a slice using some OS images, it’s really expensive to move the images to a physical directory representing the experiment.

iRODS supports both approaches. It natively supports BagIt –The preferred format for transfer of digital content to the Library of Congress. lt’s implemented as a micro‐service that packages an existing iRODS collection (logical or physical) as a bag. It can be invoked through an interactive rule or it can be invoked from a preservation policy.

 ***8. Provide multiple interfaces (including icommand, restful http, and web) to allow authenticated user to view, search and curate their objects (artifacts)  (GEC15)***

RESTful interface is in evaluation. Other than icommands, web browser, iRODS also provides GUI (iDROP) and iRODS Explorer for Windows.

 ***9. Provide interface to allow user to define an object to be archived (where the object may range from a large directory to a single file), include a descriptor (following the GENI descriptor schema), assign a persistent Digital Object Identifier (DOI, or "handle"), and decide when to push it to archive service.  (GEC16)***

The major foreseeable obstacle is the integration of iRODS and Handle System. Other than that should be simple. Possibly we will implement an iCommand called ‘iarchive’, which is similar as ‘icp’ but customized to fit the need for archiving. Similar interfaces can be implemented in iDrop and/or the web browser.

 ***10. Establish an archive service that provides long-term and reliable storage, with access via a DOI from the global handle service, with access following a local policy included in archived object. (Include at least two policies: give the object to anyone, or give the object only to its owner.)   (GEC17)***

The design and prototyping can be easily done in the logical user name space – like Unix, we can access-control the user/group/owner.