**GENI Storage and Archive Service (GSAS):**

**Configuration of Service, Structure of Directories and Files, and Use Cases**

After review on 102412 with Giridhar

Revised 112612

Revised 122112

Revised 122712 after review with Giridhar, Jeanne on 122112

Revised 022213 after discussions with Shu about BagIt

Revised 041913 after review with Giridhar, Shu and Jeanne on 030813

Revised 041913 after review with Giridhar on 032013 at GEC16

Revised 041913 to include mods sent by Giridhar on 041813

Revised 042613 after discussion with Giridhar on 042513

Revised 042913 after review with Giridhar, Shu and Jeanne on 042913

Revised 050713 to change title

Prepared by: Shu Huang, Giridhar Manepalli, Jeanne Ohren and Harry Mussman

Table of Contents

[1. Goals 4](#_Toc355698697)

[1.1 Goals for GENI Storage and Archive Service 4](#_Toc355698698)

[1.2 Goals for GENI ObjectDescriptor Schema 5](#_Toc355698699)

[1.3 Use of “DataCite Schema for the Publication and Citation of Research Data” 5](#_Toc355698700)

[2. Configuration of GENI Storage and Archive Service based on iRODS 6](#_Toc355698701)

[Issue 2.1: How are persistent accounts established for each user in iRODS? 8](#_Toc355698702)

[Issue 2.2: How are storage capacity limits established and enforced for each IRODS user? Are older objects (artifacts) flagged for removal? 8](#_Toc355698703)

[Issue 2.3: How are archive capacity limits established and enforced for each IRODS user? Are older objects (artifacts) flagged for removal? 8](#_Toc355698704)

[3. Use Cases to Store, Curate, Archive and Retrieve Experiment Objects in the GSAS 9](#_Toc355698705)

[4. Structure of Directories and Files in the GSAS 12](#_Toc355698706)

[4.1 Range of Structures 12](#_Toc355698707)

[4.2 User Structure 1 12](#_Toc355698708)

[4.3 User Structure 2 12](#_Toc355698709)

[4.4 User Structure 3 13](#_Toc355698710)

[4.5 Public Structure 1 13](#_Toc355698711)

[4.6 User Structure 2a 14](#_Toc355698712)

[4.7 User Structure 3a 16](#_Toc355698713)

[5. Access to Structure in GSAS 20](#_Toc355698714)

[5.1 From a User’s Experiment Management Environment 20](#_Toc355698715)

[5.2 From a Service Acting on Behalf of the User 20](#_Toc355698716)

[Issue 5.1: Where is the proxy certificate created? How is the proxy certificate transferred to the service? 21](#_Toc355698717)

[Issue 5.2: What happens if the proxy certificate expires? Is the user notified? How can they load an updated proxy certificate? 21](#_Toc355698718)

[Issue 5.3: How is the target information transferred to the service? 21](#_Toc355698719)

[Issue 5.4: How is the iticket transferred to the service? 22](#_Toc355698720)

[Issue 5.5: What happens if the iticket certificate expires? Is the user notified? How can they load an updated proxy certificate? 22](#_Toc355698721)

[Issue 5.6: How is all of this target information transferred to the service agent? 22](#_Toc355698722)

[5.3 Assuring a Unique experiment\_name 23](#_Toc355698723)

[5.4 Assuring a Unique file\_name (or directory\_name) 23](#_Toc355698724)

[6. Adding Descriptors (Metadata) 27](#_Toc355698725)

[6.1 Inserting metadata.xml Files into a User’s Directories 27](#_Toc355698726)

[Issue 6.1: Need to establish rules if there is a discrepancy in descriptors. 29](#_Toc355698727)

[Issue 6.2: Need to establish rules for changing or removing metadata.xml files. 29](#_Toc355698728)

[6.2 User Structure 2a with Descriptors 29](#_Toc355698729)

[6.3 User Structure 3a with Descriptors 33](#_Toc355698730)

[7. Searching Structure in GSAS 38](#_Toc355698731)

[Issue 7.1: When using a browser in the Experiment Management Environment (or elsewhere) to view artifacts (files and directories) in the GSAS, how will the associated descriptors (metadata) will be displayed? 38](#_Toc355698732)

[8. Creating a Bag and .tar File 39](#_Toc355698733)

[8.1 Example 2a-1: bag a Dataset Directory 39](#_Toc355698734)

[8.2 Example 2a-2: bag an Experiment Directory 43](#_Toc355698735)

[Issue 8.1: After the bag and .tar file have been created and used, is there some cleanup that should be done? 45](#_Toc355698736)

[Issue 8.2: After changes have been made to directories and files, what is the process for recreating the bag and .tar file? 45](#_Toc355698737)

[9. Archiving an Object 46](#_Toc355698738)

[9.1 Preparing to Archive an Object 46](#_Toc355698739)

[9.2 Example 2a-1: Archiving a Dataset Directory 47](#_Toc355698740)

[9.3 Example 2a-2: Archiving an Experiment Directory 49](#_Toc355698741)

[9.4 Example 2a-3: Archiving a Journal Article 50](#_Toc355698742)

[9.5 Updating an Archived Object 52](#_Toc355698743)

[10. Overview of v1.2 GENI ObjectDescriptor Schema 53](#_Toc355698744)

[Issue 10.1: Is there a way to derive the descriptors in an archive.xml file from descriptors in the other types of metadata.xml files, or at least an initial set of descriptors for the archive.xml file? 55](#_Toc355698745)

# 1. Goals

## 1.1 Goals for GENI Storage and Archive Service

1. A structured place to store all of the objects (artifacts) 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.)

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

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

4. Establish multiple federated iRODS services, starting at RENCI and UMass Amherst, and operate for GENI users (experimenters).

5. Establish persistent accounts for each user, and use icommands to store and retrieve objects (artifacts) for each user in the storage service.

6. Establish authentication for each user based on username/password, certificates, or proxy (delegated) certificates.

7. Establish a directory structure in the storage service for each user to accommodate multiple experiments, and a directory structure for each experiment to include all objects (artifacts) associated with that experiment, including one or more descriptors (metadata) within .xml files (following the GENI ObjectDescriptor Schema).

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

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

10. Establish an archive service that provides long-term and reliable storage, with public access via a DOI from the global handle service.

11. Include a search function in the archive service, so that an outside user can search for and then retrieve an object, but allow the object’s owner to disable search, so that an outside user needs the DOI of the object to retrieve it.

## 1.2 Goals for GENI ObjectDescriptor Schema

12. Useful for all types of objects, not just MeasurementData objects.

13. Keep it simple, with the minimum number of mandatory fields.

14. Where possible, values for fields should be automatically generated by Experiment Management Tools.

## 1.3 Use of “DataCite Schema for the Publication and Citation of Research Data”

15. When an object (artifact) is archived in the Archive Service with public access from the outside world via the Internet, use a DOI (handle) as a persistent identifier, and include descriptors (metadata) that follows DataCite Schema (ref)

# 2. Configuration of GENI Storage and Archive Service based on iRODS



Figure 1. GENI Storage and Archive Service (GSAS) configuration.

The GENI Storage and Archive Service (GSAS) is a structured place to store all of the objects for an experiment (not just measurement data objects), with descriptors (metadata), that is easy to access, useful for short to medium term storage, and that is searchable. Figure 1 summarizes the configuration of the GSAS. The configuration, development and deployment of the GSAS are detailed in a separate document (REF1).

The GSAS is based on the integrated Rule-Oriented Data System (iRODS), developed to manage data objects for large science projects (<https://www.irods.org/index.php/IRODS:Data_Grids,_Digital_Libraries,_Persistent_Archives,_and_Real-time_Data_Systems> ).

For the GENI project, it is expected that multiple federated iRODS services will be established, starting at RENCI and UMass Amherst, and operated for GENI users (experimenters).

Persistent accounts are established for each user (mechanism TBD), and each user has a (logical) home directory. A user is authenticated based on a username/password, a user certificate and/or a proxy (delegated) certificate, following the Grid Security Infrastructure (GSI) (<http://www.globus.org/security/overview.html>) methodology.

An authenticated user interacts with the iRODS icommand interface using an iclient, to create directories and to store and retrieve files (representing objects), in a manner analogous to commands in a Unix file system. A user is expected to establish a directory structure in iRODS to accommodate multiple experiments, and a directory structure for each experiment to organize and include all of the objects (artifacts) associated with that experiment.

An authenticated user can also interact with iRODS via a web interface to view, search and curate their directories and files. Searching is done by the iRODS CATalog (iCAT) system, which gathers information on each stored directory or file. As a user creates directories and loads files, they are expected to add descriptors (metadata) as XML files. (This follows the approach introduced by CNRI in the earlier prototype of the GENI Storage and Archive Service.) As these files are loaded into iRODS, customized iRODS “rules” will extract the descriptor information and pass it to iCAT, so that searches can be based on the information included in the descriptors.

The iRODS system also provides a single, separate public directory for use as a public archive, with public access from the Internet, via the Global Handle Resolution Service, using a Digital Object Identifier (DOI, or "handle") as a persistent identifier.

An authenticated user can use an icommand to push a file representing an “archived object” to the public archive directory, along with an archive.xml descriptor file (following the GENI ObjectDescriptor Schema) that includes a persistent Digital Object Identifier (DOI, or "handle").

In the public directory, the file representing the “archived object” is identified by its DOI.

As the archive.xml descriptor file is pushed to the public directory, a customized iRODS rule extracts the descriptive information and passes it to iCAT, so that searches in the public directory can be based on the information included in the descriptors. There is an option for the user to disable the search function, so that an outside user needs the DOI of the object to retrieve it.

A user is expected to retain a copy of the file representing an “archived object” within their own home directory, so that they can update it, and then push an updated file to the public directory, where it retains the original DOI.

Of course, a user can retain a copy of the file representing an “archived object” within their own home directory , and never push it to the public directory, if they do not wish to share it with the public.

The archived object may start with a single file contained within the user’s home directory, or it may include all of the structure and contents of a directory contained within the user’s home directory.

To archive a directory, the user uses the iRODS “bagit” rule; this rule creates a “bag”, which is a directory structure that includes all of the structure and contents in the subject directory plus additional index files. The rule also creates a separate tar file of the “bag”, and this is the file that is archived. When the tar file associated with a bag is retrieved, the original directory structure and all included objects can be reconstituted.

For example, the user may choose to bag a directory containing all of the measurements associated with a given experiment, and these can be then used by another researcher who wishes to analyze the measurements. Or, the user may also choose to bag all of the directories and files associated with an experiment, and these can then used by the original user or another user to repeat the experiment.

### Issue 2.1: How are persistent accounts established for each user in iRODS?

### Issue 2.2: How are storage capacity limits established and enforced for each IRODS user? Are older objects (artifacts) flagged for removal?

### Issue 2.3: How are archive capacity limits established and enforced for each IRODS user? Are older objects (artifacts) flagged for removal?

# 3. Use Cases to Store, Curate, Archive and Retrieve Experiment Objects in the GSAS

The GENI Storage and Archive Service (GSAS) is used by GENI experimenters to store, curate, archive and retrieve files representing experiment objects (artifacts). In addition, it is used by outside researchers to retrieve files representing experiment objects that have been placed in a public archive by GENI experimenters. Figure 2 summarizes seven expected use cases to store, curate, archive and retrieve files representing experiment objects by GENI experimenters and by outside researchers.



Figure 2. Use cases to store, curate, archive and retrieve files representing experiment objects in the GSAS.

In Use Case 1, a GENI experimenter uses tools in the Experiment Management Environment to create directories and store files in the GSAS using an iclient that interfaces with the icommand interface on the iRODS server, using icommands analogous to the commands addressing a Unix file system. As a user creates directories and loads files, they are expected to add descriptors (metadata) as XML files; these are used by the iRODS CATalog (iCAT) system, which gathers information on each stored directory or file. Authentication of the GENI experimenter is done by username/password or by the experimenter holding a user certificate and its associated private key.

In Use Case 2, a service acting on behalf of a GENI experimenter stores an object in the GSAS using an iclient that interfaces with the icommand interface on the iRODS server. For example, a service that collects measurement results may store a measurement data object after an experiment is completed; this object could be one file or it could be a directory containing multiple files. In Option 1, authentication of the service is done based on a proxy certificate, following the Grid Security Infrastructure (GSI) (<http://www.globus.org/security/overview.html>). The service holds a proxy certificate created from the experimenter’s user certificate, plus its associated private key. The iRODS server is configured to allow GSI authentication using proxy certificates. In Option 2, the authenticated experimenter has requested an iticket from the icommand interface on the iRODS server, and has passed it to a client in the service acting on behalf of a GENI experimenter. This service is itself an authenticated iRODS user, authenticated by holding a user certificate, and its associated private key. The authenticated service stores an object in the experimenter’s files in iRODS by using iput to the target directory, and by referencing the identifier in the iticket it received from the experimenter.

In Use Case 3, a GENI experimenter uses a browser in the Experiment Management Environment (or elsewhere) to view, search and curate objects (artifacts) in the GSAS, using its web interface. Searching is done using the iRODS CATalog (iCAT) system, which has gathered information on each stored directory or file. The user is able to view the descriptors (metadata) associated with each object (artifact). Authentication of the GENI experimenter is done by username/password or by the experimenter holding a user certificate and its associated private key.

In Use Case 4, a GENI experimenter uses tools in the Experiment Management Environment to create a bag from a selected directory in the GSAS plus its associated tar file, using an iclient that interfaces with the icommand interface on the iRODS server and invokes the irule command for “bagit”. The tar file holds all of the structure (directories and files) in the bag, and is a convenient way for the experimenter to hold all of that information in one object. Given the tar file, the original selected directory, including all contents, can be recreated. Authentication of the GENI experimenter is done by username/password or by the experimenter holding a user certificate, and its associated private key.

In Use Case 5, a GENI experimenter uses tools in the Experiment Management Environment to copy a tar file or other type of file (representing an object) in the GSAS from their home directory to the public archive directory, so that it can be shared with others. This is done using an iclient that interfaces with the icommand interface on the iRODS server, and calls a custom irule command for “archive”. As an experimenter pushes an object to the public archive directory, they are expected to add a descriptor (metadata) XML file for use by the iCAT system, so that searches in the public directory can be based on the information included in the descriptors. Optionally, the GENI experimenter can mark the descriptor to disable public searches. Authentication of the GENI experimenter is done by username/password or by the experimenter holding a user certificate, and its associated private key.

In Use Case 5b, a GENI experimenter uses tools in the Experiment Management Environment to copy an updated tar file or other type of file (representing an object) in the GSAS from their home directory to the public archive directory, to update an earlier version of their file that is being shared with others. This is done using an iclient that interfaces with the icommand interface on the iRODS server, and calls a custom irule command for “update\_archive”. Authentication of the GENI experimenter is done by username/password or by the experimenter holding a user certificate, and its associated private key.

In Use Case 6, a GENI experimenter uses tools in the Measurement Analysis Environment (or another environment) to retrieve an object from the GSAS, such as a measurement data object; this object could be one file or it could be a directory containing multiple files. It does this using an iclient that interfaces with the icommand interface on the iRODS server, using commands analogous to a those addressing a Unix file system. Authentication of the GENI experimenter is done by username/password or by the experimenter holding a user certificate, and its associated private key.

In Use Case 7, an outside researcher uses a browser to search for and then retrieve a file representing an object in the public directory of the GSAS, identified by a handle, via the public web interfaces on the Global Handle System and the GSAS. If the option has been set to disable public searcher for an object, it can still be retrieved if the outside researcher knows its DOI (handle). No authentication of the outside researcher is required.

# 4. Structure of Directories and Files in the GSAS

## 4.1 Range of Structures

Each user (experimenter) utilizes a hierarchical structure for the user’s directories and files in the GSAS, but they are free to pick a structure from a wide range of possible structures. Here are three possible user structures that range from more to less hierarchy.

## 4.2 User Structure 1

user (home\_directory)

project (directory)

experiment (directory)

step (directory)

artifact.ext (file)

artifact (directory)

artifact.ext (file)

## 4.3 User Structure 2

user (home\_directory)

experiment (directory)

step (directory)

artifact.ext (file)

artifact (directory)

artifact.ext (file)

## 4.4 User Structure 3

user (home\_directory)

experiment (directory)

artifact.ext (file)

artifact (directory)

artifact.ext (file)

## 4.5 Public Structure 1

The public portion of the GSAS typically has no hierarchy.

public (home directory)

artifact.ext (file)

## 4.6 User Structure 2a

Here is an example user structure with medium hierarchy, divided first by experiment, and then by step within the experiment. In some steps, some artifacts (files or directories) are identified by run, with a run\_number, e.g., request\_respec-0001 for Run = 1.

user\_1 (home directory)

experiment\_Alpha (directory) Experiment = Alpha

plan (directory) Step = plan

get\_resources (directory) Step = get\_resources

request\_rspec-0001.xml (file) Run = 1

request\_rspec-0002.xml (file) Run = 2

manifest\_rspec-0001.xml (file) Run = 1

manifest\_rspec-0002.xml (file) Run = 2

configure\_resources (directory) Step = configure\_resources

instrument\_script-0001.py (file) Run = 1

orchestrate\_experiment (directory) Step = orchestrate\_experiment

orchestrate\_script-0001.rb (file) Run = 1

collect\_measurements (directory) Step = collect\_measurements

measurements\_dataset-0001 (directory) Run = 1

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurements\_dataset-0002 (directory) Run = 2

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurements\_dataset-0003 (directory) Run = 3

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurements\_dataset-0004 (directory) Run = 4

mongo.tar Typical files from a GEMINI tools DB metadata.unis

topology.unis

analyze\_measurements (directory) Step = analyze\_measurements

analysis\_results-0001 (directory) Run = 1

artifact.ext (file)

…

artifact.ext (file)

analysis\_results-0002 (directory) Run = 2

artifact.ext (file)

…

artifact.ext (file)

analysis\_results-0003 (directory) Run = 3

artifact.ext (file)

…

artifact.ext (file)

document\_experiment (directory) Step = document\_experiment

journal\_article \_x (directory)

artifact.ext (file)

…

artifact.ext (file)

ver\_1\_ text.pdf

ver\_2\_ text.pdf

journal\_article \_y (directory)

artifact.ext (file)

…

artifact.ext (file)

ver\_1\_ text.pdf

ver\_2\_ text.pdf

ver\_3\_ text.pdf

experiment\_Beta (directory) Experiment = Beta

## 4.7 User Structure 3a

Here is an example user structure with low hierarchy, divided only by experiment.

Some artifacts (files or directories) are identified by run, with a run\_number, e.g., request\_respec-0001 for Run = 1.

user\_1 (home directory)

experiment\_Alpha (directory) Experiment = Alpha

request\_rspec-0001.xml (file) Run = 1

request\_rspec-0002.xml (file) Run = 2

manifest\_rspec-0001.xml (file) Run = 1

manifest\_rspec-0002.xml (file) Run = 2

instrument\_script-0001.py (file) Run = 1

orchestrate\_script-0001.rb (file) Run = 1

measurement\_dataset-0001 (directory) Run = 1

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurement\_dataset-0002 (directory) Run = 2

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurement\_dataset-0003 (directory) Run = 3

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurement\_dataset-0004 (directory) Run = 4

mongo.tar Typical files from a GEMINI tools DB metadata.unis

topology.unis

analysis\_results-0001 (directory) Run = 1

artifact.ext (file)

…

artifact.ext (file)

analysis\_results-0002 (directory) Run = 2

artifact.ext (file)

…

artifact.ext (file)

analysis\_results-0003 (directory) Run = 3

artifact.ext (file)

…

artifact.ext (file)

journal\_article \_x (directory)

artifact.ext (file)

…

artifact.ext (file)

ver\_1\_ text.pdf

ver\_2\_ text.pdf

journal\_article \_y (directory)

artifact.ext (file)

…

artifact.ext (file)

ver\_1\_ text.pdf

ver\_2\_ text.pdf

ver\_3\_ text.pdf

experiment\_Beta (directory) Experiment = Beta

# 5. Access to Structure in GSAS

## 5.1 From a User’s Experiment Management Environment

In Use Case 1, a GENI experimenter uses tools in the Experiment Management Environment to create directories and store files in the GSAS using an iclient that interfaces with the icommand interface on the iRODS server, using icommands analogous to the commands addressing a Unix file system.

1) The User is authenticated with either of these options:

a) user\_1 username/password

b) user\_1 user certificate, while holding private key

2) A target must be configured into iclient held by the User. For example:

a) url: iRODS\_server\_101

b) home directory: user\_1

c) current working directory (CWD): (typically) experiment\_Alpha (User Structure 3)

3) Then, the User can access the target in the GSAS using the iclient.

Note that, for each User, each experiment\_name (i.e., experiment\_Alpha) must be unique. If this were not true, artifacts from experiments could be overwritten or be mixed together. See Section 5.3 (below) for options to assure unique experiment\_names

## 5.2 From a Service Acting on Behalf of the User

In Use Case 2, a service acting on behalf of a GENI experimenter stores an object (artifact) in the GSAS using an iclient that interfaces with the icommand interface on the iRODS server. For example, a service that collects measurement results may store a measurement data object after an experiment is completed; this object could be one file or it could be a directory containing multiple files. There are two options for authentication:

In Option 1, authentication of the service is done based on a proxy certificate, following the Grid Security Infrastructure (GSI) (<http://www.globus.org/security/overview.html>).

1) The Service is authenticated with:

a) user\_1 proxy certificate, created from the experimenter’s user certificate, plus its associated private key. while holding associated private key

2) A target must be configured into iclient in the Service. For example:

a) url: iRODS\_server\_101

b) home directory : user\_1

c) current working directory (CWD): (typically) experiment\_Alpha (User Structure 3)

3) Then, the iclient in the Service can act on behalf of the user, and write to the target in the GSAS.

Note that when a Service uses a script to write a file (or make a directory and write a file), the file (or directory) names must be unique, and yet it is desirable if the script does this in an automatic fashion. See Section 5.4 (below) for options to assure unique file (or directory) names.

### Issue 5.1: Where is the proxy certificate created? How is the proxy certificate transferred to the service?

### Issue 5.2: What happens if the proxy certificate expires? Is the user notified? How can they load an updated proxy certificate?

### Issue 5.3: How is the target information transferred to the service?

In Option 2, the authenticated experimenter has requested an iticket from the icommand interface on the iRODS server, and has passed it to an agent in the service acting on behalf of a GENI experimenter. This service is itself an authenticated iRODS user, authenticated by holding a user certificate, and its associated private key. The authenticated service stores an object in the experimenter’s files in iRODS by using iput to the target directory, and by referencing the identifier in the iticket it received from the experimenter.

1) The User is authenticated with either of these options:

a) user\_1 username/password

b) user\_1 user certificate, while holding private key

2) The User decides on the target, for example:

a) current working directory (CWD): experiment\_Alpha (User Structure 3)

3) The authenticated User requests an iticket from iRODS with a target of their current working directory, and a limited time period:

a) request iticket: target of user\_1, CWD; expires in 2 days; returned: string xxxxxxxxx

4) The authenticated User passes the iticket to the Service agent, on behalf of the User:

a) For use by Service agent, on behalf of user\_1

b) iticket: string xxxxxxxxx; target of user\_1, CWD; expires in 2 days

5) The User must configure the target into iclient within Service, and this must match the target in the iticket. For example, the Service agent iclient is set to:

a) url: iRODS\_server\_101

b) home directory: user\_1

c) current working directory (CWD): (typically) experiment\_Alpha (User Structure 3)

6) The Service agent iclient can now act on behalf of the user, and write to the target in the GSAS.

Note that when a Service uses a script to write a file (or make a directory and write a file), the file (or directory) names must be unique, and yet it is desirable if the script does this in an automatic fashion. See Section 5.4 (below) for options to assure unique file (or directory) names.

### Issue 5.4: How is the iticket transferred to the service?

### Issue 5.5: What happens if the iticket certificate expires? Is the user notified? How can they load an updated proxy certificate?

### Issue 5.6: How is all of this target information transferred to the service agent?

## 5.3 Assuring a Unique experiment\_name

For a given User, each experiment\_name (i.e., experiment\_Alpha) must be unique. If this were not true, artifacts from experiments would be overwritten or be mixed together.

Possible methods to assure unique experiment\_names:

1) Rely upon user (experimenter) to pick a unique name.

2) During an early step of the experiment, concatenate an experiment\_name with an index that is guaranteed to change with time.

For example, method 2 this is similar to the method proposed to separate datasets in a shared OML server; in that method (ref), the index can be a Unix timestamp (e.g., 2654111326).

Then, the complete name would be: experiment\_Alpha-2654111326

This has the advantage in that it would follow the method used in a shared OML server, and the complete name could simply be carried forward.

Also, by simply comparing the timestamp values, it is easy to determine which version of experiment\_Alpha followed another.

## 5.4 Assuring a Unique file\_name (or directory\_name)

When a Service uses a script to write a file (or make a directory and write a file), the file (or directory) names must be unique, and yet it is desirable if the script does this in an automatic fashion.

Assume User Structure 3:

user\_1 (home directory)

experiment\_Alpha (directory) Experiment = Alpha

request\_rspec-0001.xml (file) Run = 1

request\_rspec-0002.xml (file) Run = 2

manifest\_rspec-0001.xml (file) Run = 1

manifest\_rspec-0002.xml (file) Run = 2

instrument\_script-0001.py (file) Run = 1

orchestrate\_script-0001.rb (file) Run = 1

measurement\_dataset-0001 (directory) Run = 1

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurement\_dataset-0002 (directory) Run = 2

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurement\_dataset-0003 (directory) Run = 3

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurement\_dataset-0004 (directory) Run = 4

mongo.tar Typical files from a GEMINI tools DB metadata.unis

topology.unis

Assume that the target configured into Service agent iclient includes:

a) url: iRODS\_server\_101

b) home directory : user\_1

c) current working directory (CWD): experiment\_Alpha

Here are three methods to assure unique file (or directory) names:

Method 1: Include a numerical index in the file or directory name, start with a known value, and increment it after each write:

Assume that when the Service agent is initialized, the target is configured into iclient, and the index is set to 0001.

This is the approach shown above for manifest\_respec (file) and for dataset (directory):

manifest\_rspec-0001.xml (file) Run = 1

manifest\_rspec-0002.xml (file) Run = 2

measurement\_dataset-0001 (directory) Run = 1

measurement\_dataset-0002 (directory) Run = 2

measurement\_dataset-0003 (directory) Run = 3

measurement\_dataset-0004 (directory) Run = 4

Method 2: Include a numerical index in the file or directory name that is known to change with time, e.g., Unix\_timestamp:

manifest\_rspec-2654111326.xml (file) Run = 1

manifest\_rspec-2654122314.xml (file) Run = 2

measurement\_dataset-2654155609 (directory) Run = 1

measurement\_dataset-2654166708 (directory) Run = 2

measurement\_dataset-2654188423 (directory) Run = 3

measurement\_dataset-2654199123 (directory) Run = 4

This timestamp in this method gives an easy way to identify and order the files (or directories), but the exact timestamp is then needed to retrieve a specific file, which requires the user to search and then pick the desired file (or directory).

Method 3: Have iRODS apply a suffix (x) to the filename when there is a repeated write, starting with (2), and then increment it after each additional write:

manifest\_rspec.xml (file) Run = 1

manifest\_rspec(2).xml (file) Run = 2

measurement\_dataset (directory) Run = 1

measurement\_dataset(2) (directory) Run = 2

measurement\_dataset(3) (directory) Run = 3

measurement\_dataset(4) (directory) Run = 4

The suffix in this method gives an easy way to identify and order the files (or directories), but the suffix is then needed to retrieve a specific file, which requires the user to search and then pick the desired file (or directory).

Note that when it is decided to use Method 1 or Method 2, it is always desirable to enable Method 3. Then, if an index is not properly incremented, iRODS will apply the suffix to distinguish the written files (directories).

6. Adding Descriptors (Metadata)

## 6.1 Inserting metadata.xml Files into a User’s Directories

To add descriptors, the user (experimenter) inserts metadata.xml files into directories.

Per the v1.2 GENI ObjectDescriptor Schema, there are five types of metadata.xml files, that each provide information for a limited context. These are the default names of these metadata.xml files:

project.xml

experiment.xml

step.xml

artifact.xml

archive.xml

In a user’s structure with hierarchy, such as User Structure 1, there is typically a one-to-one mapping between directories and types of metadata.xml files. For example:

User Structure 1

project (directory) project.xml

experiment (directory) experiment.xml

step (directory) step.xml

artifact (directory) artifact.xml

In user structures with less hierarchy, multiple types of metadata.xml files can be inserted into certain directories. For example:

User Structure 2

experiment (directory) project.xml

experiment.xml

step (directory) step.xml

artifact (directory) artifact.xml

User Structure 3

experiment (directory) project.xml,

experiment.xml

artifact (directory) step.xml

artifact.xml

User Structure 3 (alternate, where only one type of metadata.xml file (artifact.xml) needs to be associated with an artifact (directory or file))

experiment (directory) project.xml,

experiment.xml

step.xml

artifact (directory) artifact.xml

There are two possible cases:

Case 1: If a metadata.xml file is inserted into a directory with its default name (e.g., experiment.xml), the descriptive information it provides applies to that directory, and also to all subdirectories and to all included files.

As a user (experimenter) inserts a metadata.xml file into a directory, a custom rule in iRODS transfers the information to iCAT, so that directories and files may be searched based on the descriptors.

More than one type of metadata.xml files can be inserted into a single directory, and descriptive information from all are transferred to iCAT.

Case 2: If a metadata.xml file is inserted into a directory with a name that includes the name of a file in the directory, (e.g., mongo-artifact.xml in a directory with mongo.tar), the descriptive information it provides applies only to that file.

When a user (experimenter) inserts the filename-metadata.xml file into a directory, a custom rule in iRODS transfers the information to iCAT, so that the file with filename.ext (e.g., mongo.tar) may be searched based on the descriptors.

More than one type of filename-metadata.xml files can be inserted into a directory, for association with a filename.ext; the descriptive information from all are transferred to iCAT for filename.ext.

### Issue 6.1: Need to establish rules if there is a discrepancy in descriptors.

### Issue 6.2: Need to establish rules for changing or removing metadata.xml files.

Note that when a metadata.xml file is chnaged using an iclient, it is convenient to use the irsync command.

## 6.2 User Structure 2a with Descriptors

Here is the Example User Structure 2a, with representative metadata.xml included, using both default names and artifact-specific names.

user\_1 (home directory)

experiment\_Alpha (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

plan (directory)

step.xml Step = plan

get\_resources (directory)

step.xml Step = get\_resources; slice = yyyy

request\_rspec-0001.xml (file)

request\_rspec-0001-step.xml Run = 1

request\_rspec-0001-artifact.xml Artifact = request\_rspec

request\_rspec-0002.xml (file)

request\_rspec-0002-step.xml Run = 2

request\_rspec-0002-artifact.xml Artifact = request\_rspec

manifest\_rspec-0001.xml (file)

manifest\_rspec-0001-step.xml Run = 1

manifest \_rspec-0001-artifact.xml Artifact = manifest\_rspec

manifest\_rspec-0002.xml (file)

manifest \_rspec-0002-step.xml Run = 2

manifest \_rspec-0002-artifact.xml Artifact = manifest\_rspec

configure\_resources (directory)

step.xml Step = configure\_resources

instrumentize\_script-0001.py (file)

instrumentize\_scripte-0001-step.xml Run = 1

instrumentize\_scripte-0001-artifact.xml Artifact = instrumentize\_script

orchestrate\_experiment (directory)

step.xml Step = orchestrate\_experiment

orchestrate\_script-0001.rb (file)

orchestrate\_script-0001-step.xml Run = 1

orchestrate\_script-0001-artifact.xml Artifact = orchestrate\_script

collect\_measurements (directory)

step.xml Step = collect\_measurements

measurement\_dataset\_0001 (directory)

step.xml Run = 1

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

measurement\_dataset\_0002 (directory)

step.xml Run = 2

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

measurement\_dataset\_0003 (directory)

step.xml Run = 3

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

measurement\_dataset\_0004 (directory)

step.xml Run = 4

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

analyze\_measurements (directory)

step.xml Step = analyze\_measurements

analysis\_results\_0001 (directory)

step.xml Run = 1

artifact.xml Artifact = analysis\_dataset

artifact.ext (file)

…

artifact.ext (file)

analysis\_results\_0002 (directory)

step.xml Run = 2

artifact.xml Artifact = analysis\_dataset

artifact.ext (file)

…

artifact.ext (file)

analysis\_results\_0003 (directory)

step.xml Run = 3

artifact.xml Artifact = analysis\_dataset

artifact.ext (file)

…

artifact.ext (file)

document\_experiment (directory)

step.xml Step = document\_experiment

journal\_article \_x (directory)

artifact.xml Artifact = journal\_article\_fileset

artifact.ext (file)

…

artifact.ext (file)

ver\_1\_ text.pdf

ver\_1\_ text-artifact.xml Artifact = journal\_article\_text

ver\_2\_ text.pdf

ver\_2\_ text-artifact.xml Artifact = journal\_article\_text

journal\_article \_y (directory)

artifact.xml Artifact = journal\_article\_fileset

artifact.ext (file)

…

artifact.ext (file)

ver\_1\_ text.pdf

ver\_1\_ text-artifact.xml Artifact = journal\_article\_text

ver\_2\_ text.pdf

ver\_2\_ text-artifact.xml Artifact = journal\_article\_text

ver\_3\_ text.pdf

ver\_3\_ text-artifact.xml Artifact = journal\_article\_text

experiment\_Beta (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Beta

## 6.3 User Structure 3a with Descriptors

Here is the Example User Structure 3a, with representative metadata.xml included, using both default names and artifact-specific names.

user\_1 (home directory)

experiment\_Alpha (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

request\_rspec-0001.xml (file)

request\_rspec-0001-step.xml Step = get\_resources; slice = yyyy; run = 1

request\_rspec-1-artifact.xml Artifact = request\_rspec

request\_rspec-0002.xml (file)

request\_rspec-0002-step.xml Step = get\_resources; slice = yyyy; run = 2

request\_rspec-0002-artifact.xml Artifact = request\_rspec

manifest\_rspec-0001.xml (file)

manifest\_rspec-0001-step.xml Step = get\_resources; slice = yyyy; run = 1

manifest \_rspec-0001-artifact.xml Artifact = manifest\_rspec

manifest\_rspec-0002.xml (file)

manifest \_rspec-0002-step.xml Step = get\_resources; slice = yyyy; run = 2

manifest \_rspec-0002-artifact.xml Artifact = manifest\_rspec

instrumentize\_script-0001.py (file)

instrumentize\_script-0001-step.xml Step = configure\_resources; run = 1

instrumentize\_script-0001-artifact.xml Artifact = instrumentize\_script

orchestrate\_script-0001.rb (file)

orchestrate\_script-0001-step.xml Step = orchestrate\_experiment; run = 1

orchestrate\_script-0001-artifact.xml Artifact = orchestrate\_script

measurement\_dataset\_0001 (directory)

step.xml Step = collect\_measurements; run = 1

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

measurement\_dataset\_0002 (directory)

step.xml Step = collect\_measurements; run = 2

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

measurement\_dataset\_0003 (directory)

step.xml Step = collect\_measurements; run = 3

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

measurement\_dataset\_0004 (directory)

step.xml Step = collect\_measurements; run = 4

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

analysis\_results\_0001 (directory)

step.xml Step = analyze\_measurements; run = 1

artifact.xml Artifact = analysis\_dataset

artifact.ext (file)

…

artifact.ext (file)

analysis\_results\_0002 (directory)

step.xml Step = analyze\_measurements; run = 2

artifact.xml Artifact = analysis\_dataset

artifact.ext (file)

…

artifact.ext (file)

analysis\_results\_0003 (directory)

step.xml Step = analyze\_measurements; run = 3

artifact.xml Artifact = analysis\_dataset

artifact.ext (file)

…

artifact.ext (file)

journal\_article \_x (directory)

step.xml Step = document\_experiment

artifact.xml Artifact = journal\_article\_fileset

artifact.ext (file)

…

artifact.ext (file)

ver\_1\_ text.pdf

ver\_1\_ text-artifact.xml Artifact = journal\_article\_text

ver\_2\_ text.pdf

ver\_2\_ text-artifact.xml Artifact = journal\_article\_text

journal\_article \_y (directory)

step.xml Step = document\_experiment

artifact.xml Artifact = journal\_article\_fileset

artifact.ext (file)

…

artifact.ext (file)

ver\_1\_ text.pdf

ver\_1\_ text-artifact.xml Artifact = journal\_article\_text

ver\_2\_ text.pdf

ver\_2\_ text-artifact.xml Artifact = journal\_article\_text

ver\_3\_ text.pdf

ver\_3\_ text-artifact.xml Artifact = journal\_article\_text

experiment\_Beta (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Beta

# 7. Searching Structure in GSAS

A GENI user (experimenter) uses a browser in the Experiment Management Environment (or elsewhere) to view, search and curate artifacts (files and directories) in the GSAS, using its web interface.

When using a browser in the Experiment Management Environment (or elsewhere) to view artifacts (files and directories) in the GSAS, the associated descriptors (metadata) will be displayed.

Searching is done based on the metadata stored in iCAT, which includes the descriptor information loaded using the metadata.xml file.

For Authorization of the GENI experimenter is done by username/password or by the experimenter holding a user certificate and its associated private key.

### Issue 7.1: When using a browser in the Experiment Management Environment (or elsewhere) to view artifacts (files and directories) in the GSAS, how will the associated descriptors (metadata) will be displayed?

# 8. Creating a Bag and .tar File

A GENI user (experimenter) uses tools in the Experiment Management Environment to create a bag from a selected source directory in the GSAS, plus its associated tar file, using an iclient that interfaces with the icommand interface on the iRODS server and invokes the irule command for “bagit”.

The tar file holds all of the structure (directories and files) in the bag, and this allows the user to hold all of that information in one file. Given the tar file, the original selected source directory can be recreated. Authorization of the GENI experimenter is done by username/password or by the experimenter holding a user certificate and its associated private key.

The user may choose, for example, to bag a directory containing all of the measurements associated with a given experiment, and these can be used by another researcher who wishes to analyze the measurements. Or, the user may choose to bag all of the directories and files associated with an experiment, and these can then used by the original user or another user to repeat the experiment.

## 8.1 Example 2a-1: bag a Dataset Directory

Step 1: Pick source directory: dataset\_4 NOTE: must start with a directory, not an artifact (file)

experiment\_Alpha (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

collect\_measurements (directory)

step.xml Step = collect\_measurements

measurement\_dataset-0004 (directory) S1: PICK: source directory

step.xml Run = 4

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

analyze\_measurements (directory)

step.xml Step = analyze\_measurements

Step 2: Insert destination directory: sourcedirectoryname -bagit

experiment\_Alpha (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

collect\_measurements (directory)

step.xml Step = collect\_measurements

measurement\_dataset-0004 (directory) S1: PICK: source directory

step.xml Run = 4

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

dataset-4-bagit (directory) S2: INSERT: destination directory

analyze\_measurements (directory)

step.xml Step = analyze\_measurements

Step 3: Copy all applicable metadata.xml files into source directory NOTE: if this were not done, all of the metadata would not be reflected in the bag and the tar file, since bagit does not use the information in iCAT

NOTE: if a 2nd step.xml file is to be written into the directory, it should be copied as step(2).xml, to avoid confusion with the first step.xml file.

experiment\_Alpha (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

collect\_measurements (directory)

step.xml Step = collect\_measurements

measurement\_dataset-0004 (directory) S1: PICK: source directory

project.xml S3: COPY: Project = bbn\_2013\_001

experiment.xml S3: COPY: Experiment = Alpha

step(2).xml S3: COPY: Step = collect\_measurements

step.xml Run = 4

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurement \_dataset-0004-bagit (directory) S2: INSERT: destination directory

analyze\_measurements (directory)

step.xml Step = analyze\_measurements

Step 4: Run bagit command, to fill out destination directory NOTE: tar file now contains all of the information in the

and create tar file bag

experiment\_Alpha (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

collect\_measurements (directory)

step.xml Step = collect\_measurements

measurement \_dataset-0004 (directory) S1: PICK: source directory

project.xml S3: COPY: Project = bbn\_2013\_001

experiment.xml S3: COPY: Experiment = Alpha

step(2).xml S3: COPY: Step = collect\_measurements

step.xml Run = 4

artifact.xml Artifact = measurement\_dataset

mongo.tar Typical files from a GEMINI tools DB

metadata.unis

topology.unis

measurement \_dataset-0004-bagit (directory) S2: INSERT: destination directory

(now includes entire bag structure) S4: CREATE: entire bag structure

measurement \_dataset-0004-bagit.tar S4: CREATE: new tar file, that contains all of the information in the bag

analyze\_measurements (directory)

step.xml Step = analyze\_measurements

## 8.2 Example 2a-2: bag an Experiment Directory

Step 1: Identify source directory: experiment\_Alpha NOTE: must start with a directory, not a file

experiment\_Alpha (directory) S1: PICK: source directory

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

experiment\_Beta (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Beta

Step 2: Insert destination directory: experiment\_Alpha -bagit

experiment\_Alpha (directory) S1: PICK: source directory

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

experiment\_Alpha-bagit (directory) S2: INSERT: destination directory

experiment\_Beta (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Beta

Step 3: Copy all applicable metadata.xml files into source directory NOTE: if this were not done, all of the metadata would not be reflected in the bag and the tar file, since bagit does not use the information in iCAT

Note: in this case, there is nothing to copy

experiment\_Alpha (directory) S1: PICK: source directory

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

experiment\_Alpha-bagit (directory) S2: MAKE: destination directory

experiment\_Beta (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Beta

Step 4: Run bagit command, to fill out destination directory NOTE: tar file now contains all of the information in the

and create tar file bag

experiment\_Alpha (directory) PICK: source directory

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

experiment\_Alpha-bagit (directory) S2: INSERT: destination directory

(now includes entire bag structure) S4: CREATE: entire bag structure

experiment\_Alpha-bagit.tar S4: CREATE: new tar file, that contains all of the information in the bag

experiment\_Beta (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Beta

### Issue 8.1: After the bag and .tar file have been created and used, is there some cleanup that should be done?

### Issue 8.2: After changes have been made to directories and files, what is the process for recreating the bag and .tar file?

# 9. Archiving an Object

The iRODS system provides a separate public directory for use as a public archive, with access from the Internet, via the Global Handle Service, using a Digital Object Identifier (DOI or “handle”) as a persistent identifier

## 9.1 Preparing to Archive an Object

Case 1: If the object to be archived is a file, the user:

1) uses the local DOI (handle) minting service to assign a unique DOI to this file (representing an object);

2) adds a filename-archive.xml file, which includes the new DOI and descriptive information following the DataCite schema; and

3) uses an iclient that interfaces with the icommand interface on the iRODS server, and calls a custom irule command for “archive”, which pushes a copy of the file into the public directory, using the DOI as the filename, and loads all descriptive information from the filename-archive.xml file into iCAT for the public directory.

Case 2: If the object to be archived is a directory, the user archives the tar file created using the iRODS “bagit” rule.

In both cases, the user retains a copy of the “archived object” within their own home directory, for further use and updates.

The local DOI (handle) minting service assigns a unique DOI to a file with:

A dedicated prefix for each GENI iRODS zone; currently there is only one GENI zone and prefix: 10510.3.0.1

A suffix unique to each zone, which can be of any length. One way to generate a unique suffix is to use a non-repeating number, such as a Unix timestamp, or a hash of a Unix timestamp.

In the archive.xml file, the user specifies whether or not the object be found by an outside user searching the public directory iCAT. not, an outside user needs the object\_identifier to retrieve the object.

## 9.2 Example 2a-1: Archiving a Dataset Directory

Step 1: use the local DOI (handle) minting service to assign a unique DOI to this file (representing a dataset object).

Step 2: pick .tar file to be archived, representing dataset\_4.

Step 3: add a filename-archive.xml file, which includes the new DOI and descriptive information following the DataCite schema,

user\_1 (home directory)

experiment\_Alpha (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

collect\_measurements (directory)

step.xml Step = collect\_measurements

measurement\_dataset-0004 (directory) PICK: source directory

project.xml COPY: Project = bbn\_2013\_001

experiment.xml COPY: Experiment = Alpha

step(2).xml COPY: Step = collect\_measurements

step.xml Run = 4

artifact.xml Artifact = measurement\_dataset, GEMINI tools

mongo.tar

metadata.unis

topology.unis

measurement\_dataset-0004-bagit (directory) MAKE: destination directory

(now includes entire bag structure)

measurement\_dataset-0004-bagit.tar CREATE: new tar file, that contains all of the information in the bag

S2: PICK: file to be archived, representing dataset\_4

measurement\_dataset-0004-bagit-archive.xml S3: ADD: includes new DOI and descriptive information

for searching public directory

analyze\_measurements (directory)

step.xml Step = analyze\_measurements

Step 4: use an iclient that interfaces with the icommand interface on the iRODS server, and call a custom irule command for “archive”, which pushes a copy of the file into the public directory, using the DOI as the filename, and loads all descriptive information from the filename-archive.xml file into iCAT for the public directory.

public (home directory)

10510.3.0.1/1111.tar S4: NEW: file, and iCAT now includes all descriptor information that was in measurement\_dataset-0004-bagit-archive.xml

Note: Handle prefix 10510.3.0.1 is the prefix assigned to this GENI Zone.

The handle suffix (shown here as 1111) must be chosen to be unique. A random number (such as a timestamp), or a hash of a timestamp, could be chosen when the archive.xml file is first created.

## 9.3 Example 2a-2: Archiving an Experiment Directory

Step 1: use the local DOI (handle) minting service to assign a unique DOI to this file (representing a dataset object).

Step 2: pick .tar file to be archived, representing experiment\_Alpha.

Step 3: add a filename-archive.xml file, which includes the new DOI and descriptive information following the DataCite schema,

user\_1 (home directory)

experiment\_Alpha (directory) PICK: source directory

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

experiment\_Alpha-bagit (directory) MAKE: destination directory

(now includes entire bag structure)

experiment\_Alpha-bagit.tar CREATE: new tar file, that contains all of the information in the bag

S2: PICK: file to be archived, representing experiment\_Alpha

experiment\_Alpha-bagit-archive.xml S3: ADD: includes new DOI and descriptive information for searching public directory

experiment\_Beta (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Beta

Step 4: use an iclient that interfaces with the icommand interface on the iRODS server, and call a custom irule command for “archive”, which pushes a copy of the file into the public directory, using the DOI as the filename, and loads all descriptive information from the filename-archive.xml file into iCAT for the public directory.

public (home directory)

10510.3.0.1/2222.tar S4: NEW: file, and iCAT now includes all descriptor information that was in experiment\_Alpha-bagit-archive.xml

Note: Handle prefix 10510.3.0.1 is the prefix assigned to GENI.

The handle suffix (shown here as 2222) must be chosen to be unique. A random number (such as a timestamp), or a hash of a timestamp, could be chosen when the archive.xml file is first created.

## 9.4 Example 2a-3: Archiving a Journal Article

Step 1: use the DOI (handle) minting service to assign a unique DOI to this file

Step 2: add a filename-archive.xml file, which includes the new DOI and descriptive information following the DataCite schema,

Step 1: use the local DOI (handle) minting service to assign a unique DOI to this file.

Step 2: pick file to be archived, representing journal\_article\_x.

Step 3: add a filename-archive.xml file, which includes the new DOI and descriptive information following the DataCite schema,

user\_1 (home directory)

experiment\_Alpha (directory)

project.xml Project = bbn\_2013\_001

experiment.xml Experiment = Alpha

document\_experiment (directory)

step.xml Step = document\_experiment

journal\_article \_x (directory)

artifact.xml Artifact = journal\_article\_fileset

artifact.ext (file)

…

artifact.ext (file)

ver\_1\_ text.pdf

ver\_1\_ text-artifact.xml Artifact = journal\_article\_text

ver\_2\_ text.pdf S2: PICK: file to be archived, representing journal\_article\_x

ver\_2\_ text-artifact.xml Artifact = journal\_article\_text

ver\_2\_ text-archive.xml S3: ADD: includes new DOI and descriptive information for searching public directory

Step 4: use an iclient that interfaces with the icommand interface on the iRODS server, and call a custom irule command for “archive”, which pushes a copy of the file into the public directory, using the DOI as the filename, and loads all descriptive information from the filename-archive.xml file into iCAT for the public directory.

public (home directory)

10510.3.0.1/3333.tar S4: NEW: file, and iCAT now includes all descriptor information that was in ver\_2\_text-archive.xml

Note: Handle prefix 10510.3.0.1 is the prefix assigned to GENI.

The handle suffix (shown here as 3333) must be chosen to be unique. A random number (such as a timestamp), or a hash of a timestamp, could be chosen when the archive.xml file is first created.

## 9.5 Updating an Archived Object

A GENI experimenter uses tools in the Experiment Management Environment to copy an updated .tar file or other type of file (representing an object) in the GSAS from their home directory to the public archive directory. Only the owner of the file is given write permission in the public directory.

At the same time, the associated archive.xml file may be updated., but the DOI (handle) remains the same.

All versions of the file are retained in the public directory, but the latest version is the one shared with others.

Updating a file is done by a User with an iclient that interfaces with the icommand interface on the iRODS server, and calls a custom irule command for “update\_archive”.

Authentication of the GENI experimenter is done by username/password or by the experimenter holding a user certificate, and its associated private key.

# 10. Overview of v1.2 GENI ObjectDescriptor Schema

The v1.2 GENI ObjectDescriptor Schema is described with an .xsd file at <http://groups.geni.net/geni/wiki/GSAS#a3GENIObjectDescriptorSchema>. An overview is presented here.

This schema is the culmination of many discussions within the GENI Instrumentation and Measurement (I&M) working group, and by the authors of this document.

It is intended to be useful for all types of objects, not just MeasurementData objects.

A principal goal was to keep it simple, with the minimum number of mandatory fields, and, where possible, to have values for fields automatically generated by Experiment Management Tools.

Also, when a file (representing an object) is archived with public access, a DOI (handle) is used as a persistent identifier, and descriptors (metadata) are included that follow the DataCite Schema. The DataCite Schema is described at <http://www.dlib.org/dlib/january11/starr/01starr.html> and <http://groups.geni.net/geni/wiki/GSAS#a4DataCiteSchema>. The DataSite Schema was chosen since it was established to describe research results that are made available in the public domain.

Storage Service for user a, in zone y, on iRODS server 101:

Implicit: hierarchical structure of directories, artifact files, and associated metadata files

Implicit: permission by user within zone to read/write

Explicit: artifact files always in a directory (name/path, date last modified, size)

Explicit: metadata.txt files always in a directory (name/path, date last modified, size), so that they are associated with the directory or with individual artifact files

Multiple schemas, each for a specific context:

project.xml

experiment.xml

step.xml

artifact.xml

archive.xml

Key: mandatory [M]

optional [O]

Note: GENI-specific fields noted; others are generic

project.xml file:

GENI\_project\_identifier [M]

Project title fields [M]: (title [M]; keywords [O]; abstract [O]; notes [O])

PI fields [M]: (individual [M]; organization [M]); optional multiple entries

Project date/time interval fields [O]: (start [M]; end [M]); optional multiple entries

experiment.xml file:

GENI\_experiment\_identifier [M]

Experiment title fields [M]: (title [M]; keywords [O]; abstract [O]; notes [O])

Experimenter fields [M]: (individual [M]; organization [M]); optional multiple entries

Experiment date/time interval fields[O]: (start [M]; end [M]); optional multiple entries

step.xml file:

Step\_type fields: (primary [M, follows controlled vocabulary]; secondary [O])

Step\_sequence\_identifier [M]

Step title fields [M]: (title [M]; keywords [O]; abstract [O]; notes [O])

Step date/time interval fields [O]: (start [M]; end [M]); optional multiple entries

GENI\_resource fields: [O] (GENI resource type [M, follows controlled vocabulary]; GENI\_resource\_identifier [M]); optional repeat

Note: Step\_sequence\_identifier should be used to indicate the position of this step in the sequence of all steps, e.g., run=1 and step=5

Note: GENI\_slice\_identifier should be included as a GENI\_resource\_identifer in at least one step.xml file

artifact.xml file:

Artifact\_type fields: (primary [M, follows controlled vocabulary]; secondary [O]; version [O])

Artifact\_name fields [M]: (name [M]; keywords [O]; abstract [O]; notes [O])

Artifact\_interpretation fields: (description [M]; link\_to\_read\_me.txt [O]; version [O]; link\_to\_interpretation\_resource [O])

Artifact\_attributed\_to fields [O]: (project [O]; individual [O]; organization [O]; reference [O]; link [O]; date/time received [O])

archive.xml file:

Object\_identifier [M] (DataCite 1) Note: Starts with all 9’s, before first archive

Object\_date\_last\_archived [M] Note: Starts with all 9’s, before first archive

Object\_available\_public = Y/N [M] Note: Starts with N, before first archive

Object\_searchable\_public = Y/N [M] Note: Indicates whether object be found by search on public side; if not, need Object\_identifier to retrieve it.

Object\_creator [M] (DataCite 2)

Object\_title [M] (DataCite 3)

Object\_publisher [M] (DataCite 4)

Object\_publicationYear [M] (DataCite 5)

Object\_subject [O] (DataCite 5)

through

Object\_description [O] (DataCite 17)

### Issue 10.1: Is there a way to derive the descriptors in an archive.xml file from descriptors in the other types of metadata.xml files, or at least an initial set of descriptors for the archive.xml file?