G E N I

Global Environment for Network Innovations

PlanetLab GENI Control Framework Overview

Document ID: GENI-SE-CF-PLGO-01.2

January 14, 2009

DRAFT

Prepared by:
The GENI Project Office
BBN Technologies
10 Moulton Street
Cambridge, MA 02138 USA

Issued under NSF Cooperative Agreement CNS-0737890

TABLE OF CONTENTS

DRAFT

DRAFT

1 Document Scope

This section describes this document's purpose, its context within the overall GENI document tree, the set of related documents, and this document's revision history.

1.1 Purpose of this Document

This document provides an overview of the PlanetLab GENI control framework being implemented for Spiral 1, for use in Cluster B. It is a DRAFT, to be used for discussion in the GENI Facility Control Framework working group. (Note: A review of this document by the PlanetLab team is underway, but has not yet been completed.) It provides a description of the PlanetLab GENI control framework structure, a summary of how it meets the requirements as presented in the "GENI Control Framework Requirements", and a view of its implementation at the start and the finish of Spiral 1.

Some of the material in this document is taken from the GENI System Requirements document.

Some of the material in this document is taken from the GENI System Overview document.

Some of the material in this document is taken from the GENI Control Framework Requirements document.

Some of the material is taken from a draft "Slice-Based Facility Architecture (SFA)" document, Draft v1.02, November 3, 2008, by Larry Peterson (editor), Soner Sevinc, Jay Lepreau, Robert Ricci, John Wroclawski, Ted Faber, Stephen Schwab and Scott Baker.

Some of the material is taken from a draft "Planet Lab Implementation of the Slice-Based Facility Architecture," Draft v0.01, November 7, 2008, by Larry Peterson, Soner Sevinc and Scott Baker

Some of the material is taken from a draft "Geniwrapper Design Overview" by Scott Baker.

DRAFT

1.2 Context for this Document

Figure 1-1. below shows the context for this document within GENI's overall document tree.

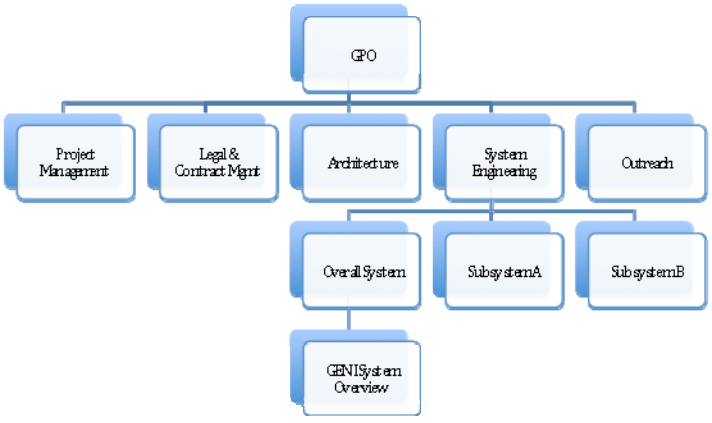

Figure 1-1. This Document within the GENI Document Tree.

1.3 Related Documents

The following documents of exact date listed are related to this document, and provide background information, requirements, etc., that are important for this document.

1.3.1 National Science Foundation (NSF) Documents

Document ID	Document Title and Issue Date
N / A	

1.3.2 GENI Documents

Document ID	Document Title and Issue Date
GENI-SE-SY-RQ-01.4	GENI System Requirements, September 18, 2008 http://www.geni.net/docs/GENI-SE-SY-RQ-01.7.pdf
GENI-SE-SY-SO-01.5	GENI System Overview, September 19, 2008, http://www.geni.net/docs/GENISysOvrvw092908.pdf
GENI-SE-CF-RQ-01.x	GENI Control Framework Requirements, November 21, 2008, http://geni.bbn.com:8080/docushare/dsweb/Services/Document-1234

1.3.3 Standards Documents

Document ID	Document Title and Issue Date
N / A	

1.3.4 Other Documents

Document ID	Document Title and Issue Date
GDD 06-10	"Towards Operational Security for GENI," by Jim Basney, Roy Campbell, Himanshu Khurana, Von Welch, GENI Design Document 06-10, July 2006. http://www.geni.net/GDD/GDD-06-10.pdf
GDD 06-23	"GENI Facility Security," by Thomas Anderson and Michael Reiter, GENI Design Document 06-23, Distributed Services Working Group, September 2006. http://www.geni.net/GDD/GDD-06-23.pdf
N/A	"GMC Specifications," edited by Ted Faber, Facility Architecture Working Group, September 2006. http://www.geni.net/wsdl.php
GDD 06-24	"GENI Distributed Services," by Thomas Anderson and Amin Vahdat, GENI Design Document 06-24, Distributed Services Working Group, November 2006. http://www.geni.net/GDD/GDD-06-24.pdf

DRAFT

GDD 06-38	"GENI Engineering Guidelines," edited by Ted Faber, GENI Design Document 06-38, Facility Architecture Working Group, December 2006. http://www.geni.net/GDD/GDD-06-38.pdf
GDD 06-42	"Using the Component and Aggregate Abstractions in the GENI Architecture," by John Wroclawski, GENI Design Document 06-42, Facility Architecture Working Group, December 2006. http://www.geni.net/GDD/GDD-06-42.pdf
N/A	"Slice Based Facility Architecture," Draft v1.02, November 3, 2008, by Larry Peterson, et.al. http://svn.planet-lab.org/attachment/wiki/GeniWrapper/sfa.pdf
N/A	"Planet Lab Implementation of the Slice-Based Facility Architecture," Draft v0.01, November 7, 2008, by Larry Peterson, Soner Sevinc and Scott Baker http://www.cs.princeton.edu/~llp/geniwrapper.pdf
N/A	"geniwrapper" and "Geniwrapper Design Overview" by Scott Baker http://svn.planet-lab.org/wiki/GeniWrapper and http://svn.planet-lab.org/wiki/OverviewLinkGoesHere
N/A	"Decentralized Trust Management," 1996, by Matt Blaze, et.al, AT&T Research. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.6276
N/A	"Compliance Checking in the PolicyMaker Trust Management System," 1998, by Matt Blaze, et.al, AT&T Research. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.2525
N/A	"The Role of Trust Management in Distributed Systems Security," 1999, by Matt Blaze, et.al, AT&T Research. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.7726

DRAFT

1.4 Document Revision History

The following table provides the revision history for this document, summarizing the date at which it was revised, who revised it, and a brief summary of the changes. This list is maintained in reverse chronological order so the newest revision comes first in the list.

Revision	Date	Revised By	Summary of Changes
01.1	12/15/08	H. Mussman	Completed draft, utilizing PlanetLab material from Larry Peterson and Scott Baker, and following structure from Control Framework Requirements document.
01.2	1/14/09	H. Mussman	Updated with small changes.
01.3			
01.4			

DRAFT

2 GENI System Overview

2.1 Major Entities and their Relationships

Figure 2-1 presents a block diagram of the GENI system covering the major entities within the overall system. Optional (but desirable) parts are shown "grayed-out." See the GENI System Overview document at http://www.geni.net/docs/GENISysOvrvw092908.pdf for more details.

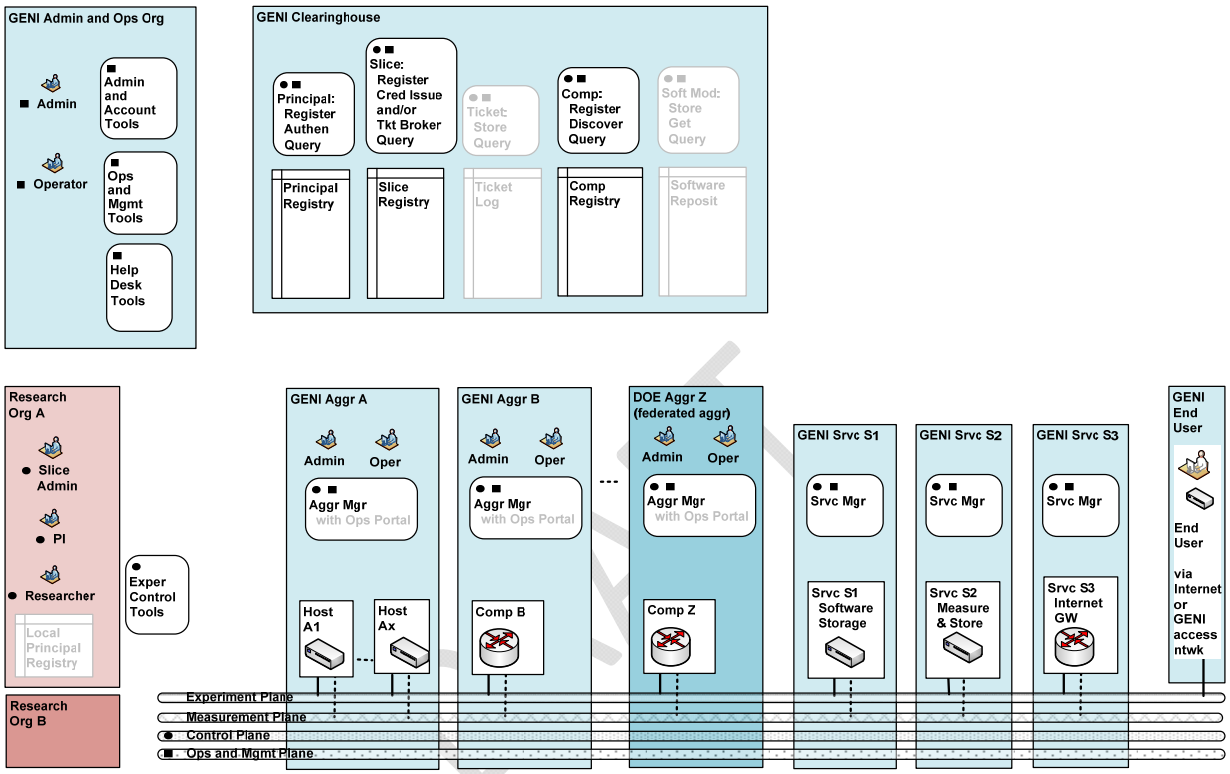

Figure 2-1. GENI System Diagram.

2.2 Federated Suites

Figure 2-2 provides a system diagram illustrating federation between one GENI suite and another. As a hypothetical example, it depicts federation between a US-based GENI suite and a compatible suite in the European Union (EU).

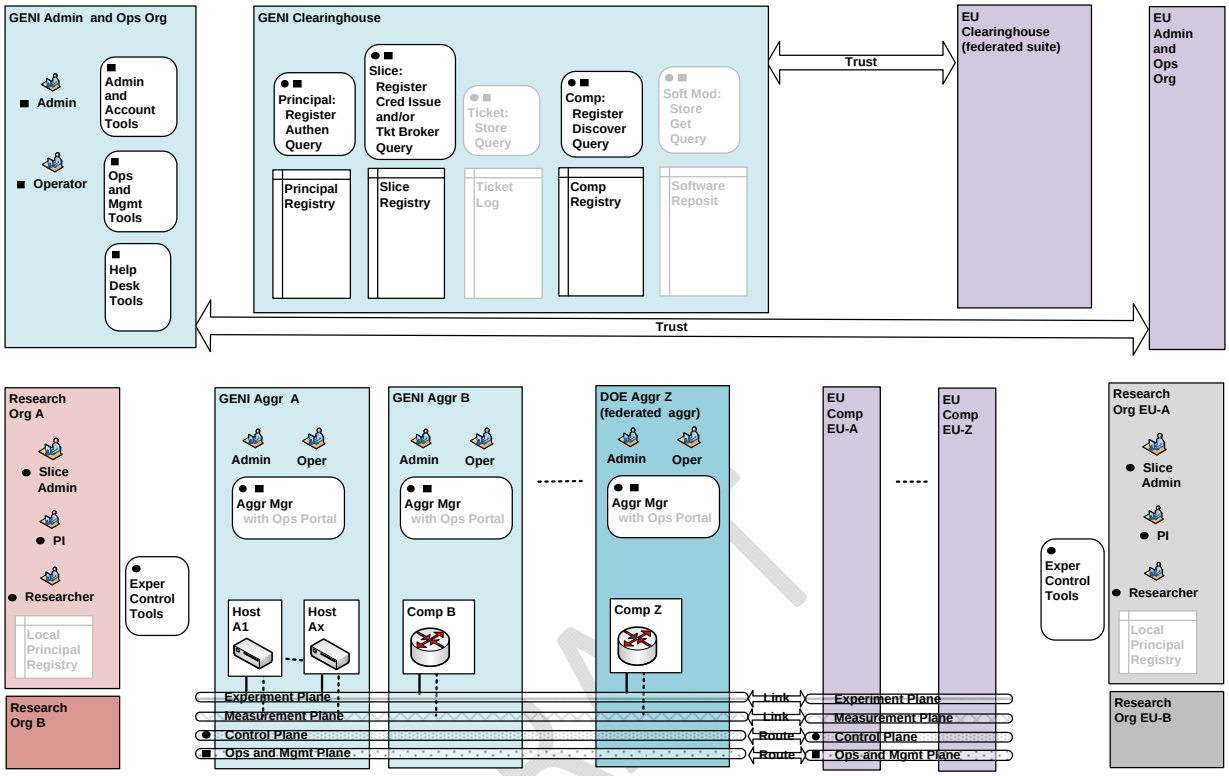

Figure 2-2. System Diagram with Federated Infrastructure Suites.

DRAFT

2.3 Slices

Figure 2-3 shows two researchers from different organizations managing their two experiments in two corresponding slices. Each slice spans an interconnected set of slivers on multiple aggregates and/or components in diverse locations. Each researcher remotely discovers, reserves, configures, programs, debugs, operates, manages, and teardowns the "slivers" that are required for their experiment. Note that the clearinghouse keeps track of these slices for troubleshooting or emergency shutdown.

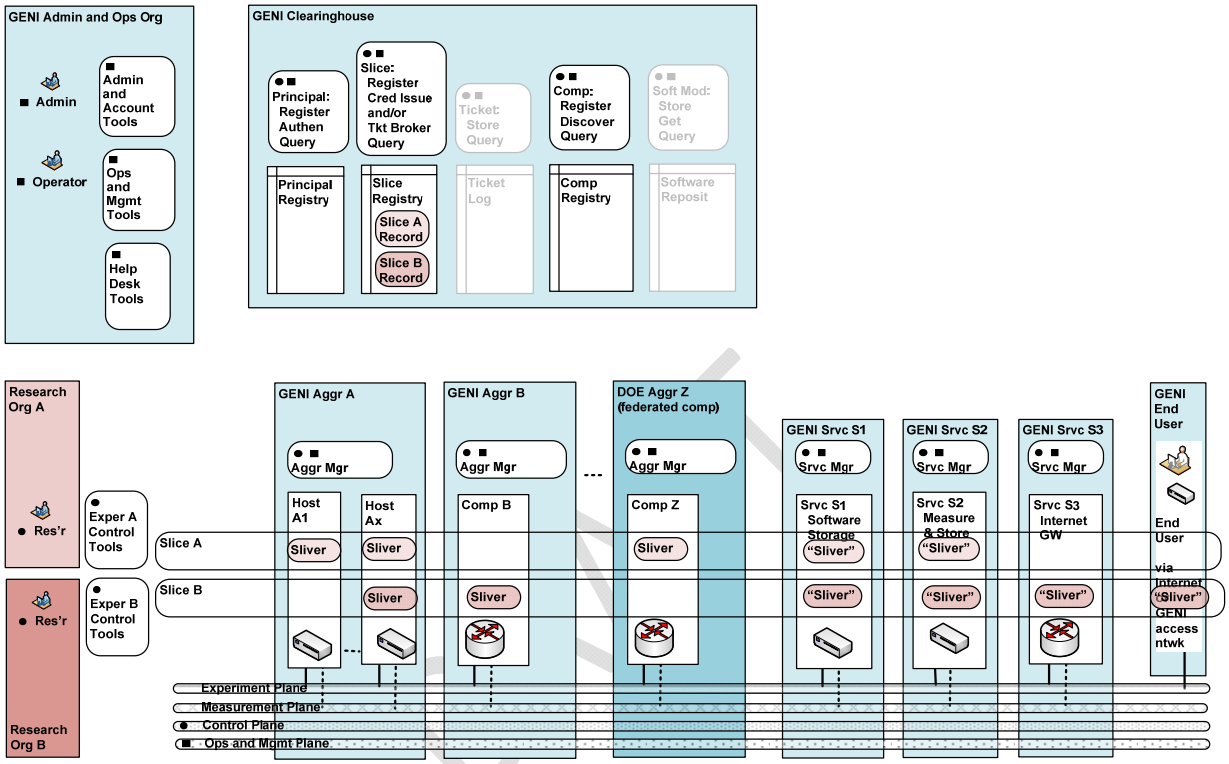

Figure 2-3. Two GENI Slices.

An aggregate manager a) interacting with the researcher (or her proxies) via the control plane and b) configuring the devices over internal interfaces establishes Slivers. Components may be virtualized, and can thus provide resources for multiple experiments at the same time, but keep the experiments isolated from one another. In addition, each slice requires its own set of experiment support services. Furthermore, as shown in Slice B, "opt-in" users may join the experiment running in a slice, and thus be associated with that slice.

3 GENI Control Framework Overview

3.1 Definition

The GENI control framework is defined in the GENI Control Framework Requirements document at http://geni.bbn.com:8080/docushare/dsweb/Services/Document-1234 .

It includes these entities:

- A "clearinghouse" consisting of principal, component and slice registries, plus related services.
- Services associated with each aggregate.
- Principals typically using tools, and acting as clients.

The GENI control framework defines:

- Interfaces between all entities.
- Planes for transporting messages between all entities.
- Message types, including basic protocols and required functions.
- Message flows necessary to realize key experiment scenarios.

3.2 Requirements

The GENI control framework requirements are presented in the GENI Control Framework Requirements document at http://geni.bbn.com:8080/docushare/dsweb/Services/Document-1234 .

3.3 Implementation Approach for Spiral 1 Prototypes

Five control framework implementations are being developed for Spiral 1 prototypes, based on the following systems and software packages:

- PlanetLab, a system that allows researchers to conduct experiments on hosts located at various sites; see http://svn.planet-lab.org/wiki/GeniWrapper and http://groups.geni.net/geni/wiki/PlanetLab .
- ProtoGENI, which is based Emulab, a system that allows researchers to conduct experiments on hosts and other equipment located in various sites; see https://www.ProtoGENI.net/trac/ProtoGENI and http://groups.geni.net/geni/wiki/ProtoGENI .
- ORCA resource allocation software; see http://nicl.cod.cs.duke.edu/orca/ and http://groups.geni.net/geni/wiki/ORCABEN .
- ORBIT, a system that allows researchers to conduct experiments on a federated arrangement of wireless networks, utilizing peridically disconnected resources; see http://www.orbit-lab.org/wiki/WikiStart and http://groups.geni.net/geni/wiki/ORBIT .
- TIED, a system that allows researchers to conduct experiments on a federated arrangement of hosts located in various sites; see http://seer.isi.deterlab.net/ and http://groups.geni.net/geni/wiki/TIED .

Each control framework will be used for one cluster of projects, and typically provides the clearinghouse and a reference implementation of the aggregate manager for its cluster. Researchers in a given cluster will typically be able to conduct experiments only on the prototype aggregates and components within that cluster.

DRAFT

At the completion of Spiral 1, these control framework prototypes will be compared and evaluated.

For Spiral 2 prototyping during the following year, improvements and/or consolidations are expected. Useful features in one control framework may be adopted by another. A particular project may choose to move from one control framework to another. And, it is also possible that two (or more) control frameworks may merge, or possibly just federate.

Sections 4 though 11 present an overview of the PlanetLab GENI-based control framework implementation.

DRAFT

4 PlanetLab GENI Control Framework Structure

A block diagram of the basic PlanetLab GENI control framework structure is shown in Figure 4-1, which includes:

- Principal, Slice and Component Registries, which are the key entities in the Clearinghouse, and which include, respectively, Principal, Slice and Component Registry Services, plus Principal, Slice and Management Authority Services.
- One or more Aggregates which include Components.
- Principals, such as Administrators, PIs, Operators, and Researchers.
- One or more Slice Managers which are used by the Researchers to setup and manage slices.

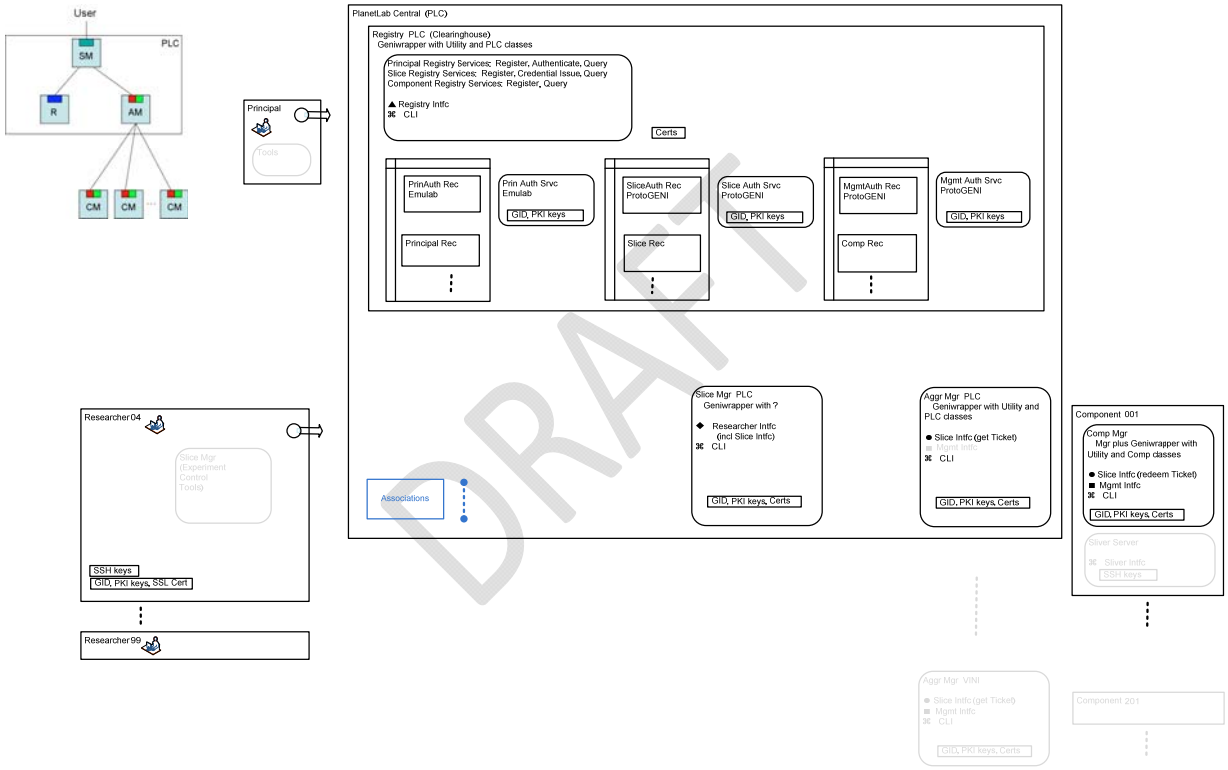

Figure 4-1. PlanetLab GENI Control Framework Structure.

The PlanetLab GENI control framework structure is based on the “Slice-based Facility Architecture” (SFA) at http://svn.planet-lab.org/attachment/wiki/GeniWrapper/sfa.pdf . Note that the SFA is based on earlier GENI efforts, including: "GMC Specifications," "GENI Distributed Services," and "GENI Engineering Guidelines."

This control framework structure is described in the “Planet Lab Implementation of the Slice-Based Facility Architecture,” at http://www.cs.princeton.edu/~llp/geniwrapper.pdf . The current implementation starts with PlanetLab Central and adds the geniwrapper software module (utility and PLC classes) to provide the following entities shown in Figure 4-1:

- The PLC Registry that realizes the Principal, Slice and Component Registries and their included services.
- The PLC Aggregate Manager.
- The PLC Slice Manager.

Furthermore, the geniwrapper software module (utility and component classes) is utilized to build:

- Each Component Manager

This current design of the geniwrapper software module is summarized in http://svn.planet-lab.org/wiki/GeniWrapper and http://svn.planet-lab.org/wiki/OverviewLinkGoesHere .

The **`geniwrapper`** module is distributed as part of the MyPLC software package, and is independently available at **`http://svn.planet-lab.org`**.

Note that the **`geniwrapper`** module can be accessed in isolation, e.g., to provide a starting point for building a stand-alone AM or CM.

The **`geniwrapper`** module defines the following sets of classes:

- Utility classes that implement GIDs, credentials, and tickets, as well as an underlying secure remote invocation mechanism. These classes are contained in the **`/util`** directory of the module.
- PLC classes that implement the registry, slice, and management interfaces exported by the Aggregate Manager (AM) and Registry (R) co-located with PLC. These classes are contained in the **`/plc`** directory of the module.
- Component classes that implement the slice and management interfaces exported by the Component Manger (CM) co-located with the node manager of each node. These classes are contained in the **`/component`** directory of the module.

The module also includes a command-line client program can be used to exercise the AM, CM, and Registry servers. The command-line client is contained in the **`/cmdline`** directory of the module.

Two files, geniserver.py and geniclient.py implement a basic Geni server and client.

Geniserver forms the basis of any server that exports a Geni interface. Examples include the PLC and Component wrappers. The Geniserver class itself does not export any useful API functions other than a "noop" function that can be used to test the server interface. Descendant classes register additional API functions by overriding the register_function() member of the geniserver object.

Geniserver provides a function, decode_authentication, that decodes credentials. Credentials are supplied as the first parameter to many registry and slice interface API functions. This function converts the credential string supplied by the user into a credential object, checks to see that the key the caller is using to encrypt the SSL connection matches the public key in the caller GID of the credential, checks to see that the credential allows the operation the caller is attempting to do, and finally verifies that the parentage of the credential traces back to a trusted root.

Geniclient provides a variety of client-side stubs for invoking operations on GENI interfaces. These stubs convert objects into strings that may be encoded by XMLRPC, call the associated XMLRPC function, and convert the results back into objects. Use of the Geniclient class is optional, but it makes a convenient mechanism to execute API calls.

DRAFT

4.1 Registries

The PLC Registry realizes the Principal, Slice and Component Registries, which hold all of the records plus associated services which are necessary for the operation of the PlanetLab GENI suite.

The PLC Registry exports a Registry Interface as defined in the SFA document, but it is implemented on top of the PlanetLab database.

In the PLC Registry, each registry record is given by the 4-tuple (Name, GID, Type, Info), where:

Name specifies the Human Readable Name (HRN) of the object

GID is the GID of the object

Type is user | sa | ma | slice | component

Info is comprised of the following sub-fields:

Pointer is a pointer to the record in the PL database

pl_info is planetlab-specific info (when talking to client)

geni_info = geni-specific info (when talking to client)

The pointer is interpreted depending on the type of the record. For example, if the type=="user", then pointer is assumed to be a person_id that indexes into the persons table. A given HRN may have more than one record, provided that the records are of different types. For example, planetlab.us.arizona may have both an SA and a MA record, but cannot have two SA records.

Per the SFA, the basic functionality of a registry is to map HRNs into records. However, because of the interactions between geniwrapper and PLC, the registry does more than act as a simple database. The registry performs API calls on PLC that create slices, sites, users, etc., and as such may indirectly cause slices to be instantiated on components, because components are also linked to PLC.

The mapping of GENI objects to PlanetLab objects is relatively straightforward:

slice = slice

user = person

component = node

sa = site

ma = site

The one part that is slightly counterintuitive is SA and MA, which both map to the PlanetLab site object. In a unified registry (a registry that serves as both slice and component registry), these will map to the same site record in the PLC database. However, there are two distinct GENI records, one for the SA and one for the MA.

Per the SFA, each registered object has a Global Identifier (GID) that includes a UUID and the object's Public Key. The UUID is a random number, generated following X.667 (RFC4122), that is guaranteed to be unique.

The PLC Registry includes two command-line interfaces.

The End-User CLI (not yet completed) allows users to walk and update the registry, as well as to create and control slices. The main command -- sfi, for Slice-based Facility Interface -- hides details of credentials, GID, and remote servers. The following outlines the sub-commands we plan to support:

Walk and update the Registry

Learn about Nodes and Slices

DRAFT

Create and control slices

The Developer CLI is located in the cmdline directory and can be invoked by running genicli.py. Specifying "genicli.py help" will display a list of available commands. Several examples of using the CLI are presented in the form of shell scripts in the cmdline directory. These scripts demonstrate creating slices, authorities, users, nodes, and getting tickets and redeeming tickets.

DRAFT

4.2 Aggregates and Components

The basic PlanetLab GENI suite includes one aggregate, the PLC Aggregate, whose Aggregate Manager is located in PLC. It is realized with the geniwrapper software module (utility and PLC classes) that is built on top of PLC. See Figure 4.2

Many Components (nodes) are typically associated with the PLC Aggregate Manager. Each includes a Component Manager that is realized with the geniwrapper software module (utility and Component classes) combined with other component management processes.

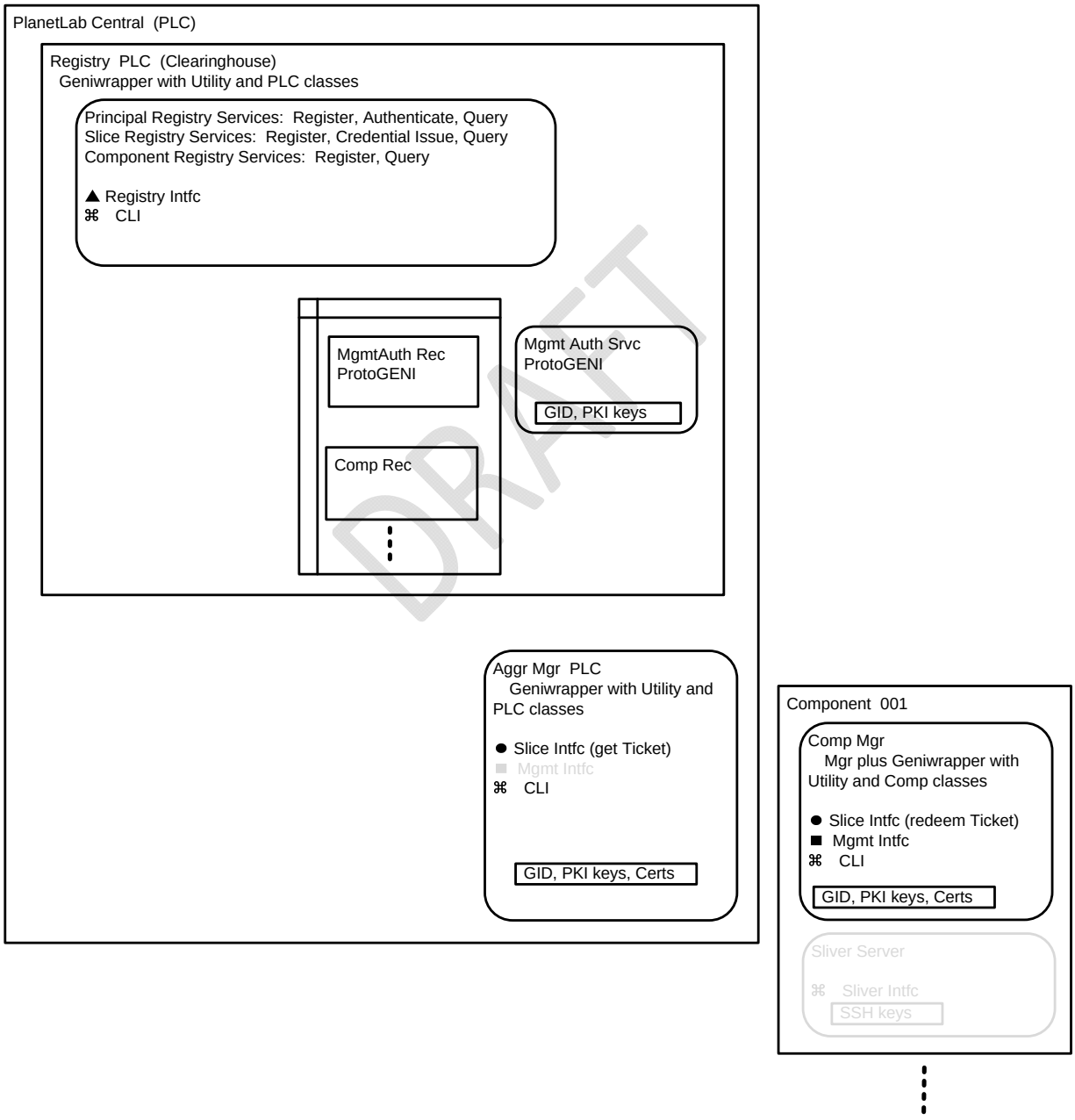

Figure 4-2. The PLC Aggregate and its Components.

The PLC Aggregate Manager exports the Slice Interface defined in the SFA document.

Since it is implemented on top of the PLCAPI, and as such, it can be used to create, terminate, and control slices on a PlanetLab-wide basis. It can also be used to get a Ticket that can then be subsequently redeemed at a CM running on any PlanetLab node managed by this instance of PLC.

Since it is built using the geniwrapper software modules, the PLC Aggregate Manager also exports the End-User and Developer CLIs; see Section 4.1

Each Component Manager implements the Slice and (component) Management Interfaces as defined in the SFA.

It includes functions for redeeming Tickets, starting/stopping/resetting/deleting slices, and management such as rebooting the component. Currently, slice control operations invoked on a CM are successful only if the slice was created using a ticket (as opposed to created by invoking InstantitateSlice on the PLC aggregate).

Since it is built using the geniwrapper software modules, the Component Manager also exports the End-User and Developer CLIs; see Section 4.1

Note that mainpulating tickets is split between the PLC wrapper in the Aggregate Manager and the Component wrapper in the Component Manager. Specifically, the authoritative copy of planetlab state is stored on PLC and only cached on the components. Thus, GetTicket?() is implemented by the PLC wrapper in the Aggregate Manager, and RedeemTicket?() is implemented by the Component wrapper in the Component Manager. Attempting to call GetTicket?() on a component will fail.

Note that additional aggregates (with associated components) can be defined; see one proposed approach outlined in Section 10.2.

DRAFT

4.3 Principals

A PlanetLab GENI principal (user) can be:

- A principal (user) acting from a server utilizing a browser.
- A principal (user) acting from a server utilizing a set of helper tools, such as a Researcher utilizing a local Slice Manager (Experiment Control Tools).

A principal in PlanetLab GENI has privileges to allow it to play one (or more than one) of the following roles in the PlanetLab GENI suite:

- Principal administrators, who act for the PlanetLab GENI suite or a research organization, and are responsible for principal records and the authentication of principals.
- Aggregate administrators, who act for the PlanetLab GENI suite or an owning organization, and are responsible for aggregate records.
- Slice administrators, who act for the PlanetLab GENI suite or a research organization, and are responsible for slice records.
- PIs, who act for a research organization, and are responsible for slice records, the researchers assigned to a slice, and for managing slices, including all of their slivers.
- Operators, who act for the PlanetLab GENI suite or an owning organization, and are responsible for operations and management functions within an aggregate (or component).

- Researchers, who utilize the PlanetLab GENI suite for running experiments, deploying experimental services, etc.

4.4 Services

A Service can be deployed in the GENI suite to assist in the setup and running of experiments (or in the setup and running of the GENI suite). The PlanetLab GENI suite utilizes a Slice Manager service that provides many of the functions envisioned for a GENI Experiment Control service.

In the basic PlanetLab GENI suite shown in Figure 4-1, a basic (trivial) Slice Manager is to be realized using PLC and the geniwrapper software module. A prototype implementation is in progress. It will export the slice interface, and in turn, call the slice interface on the set of aggregates with which this instance of PLC peers. Typically, the SM will call each peer AM and ask for the set of available components (and subsequently display the union of these lists to users), and later, call the appropriate AMs to instantiate a slice on the components managed by that AM. The SM will maintain a database of all slices created by behalf of users, including a record of where those slices have been instantiated. Researchers (users) interact with this slice manager to create and control their slices.

Researchers (users) may also utilize an alternative Slice Manager. The alternative slice manager could be located in a centralized location, or it could be dedicated to one Researcher as shown in Figure 4-3.

DRAFT

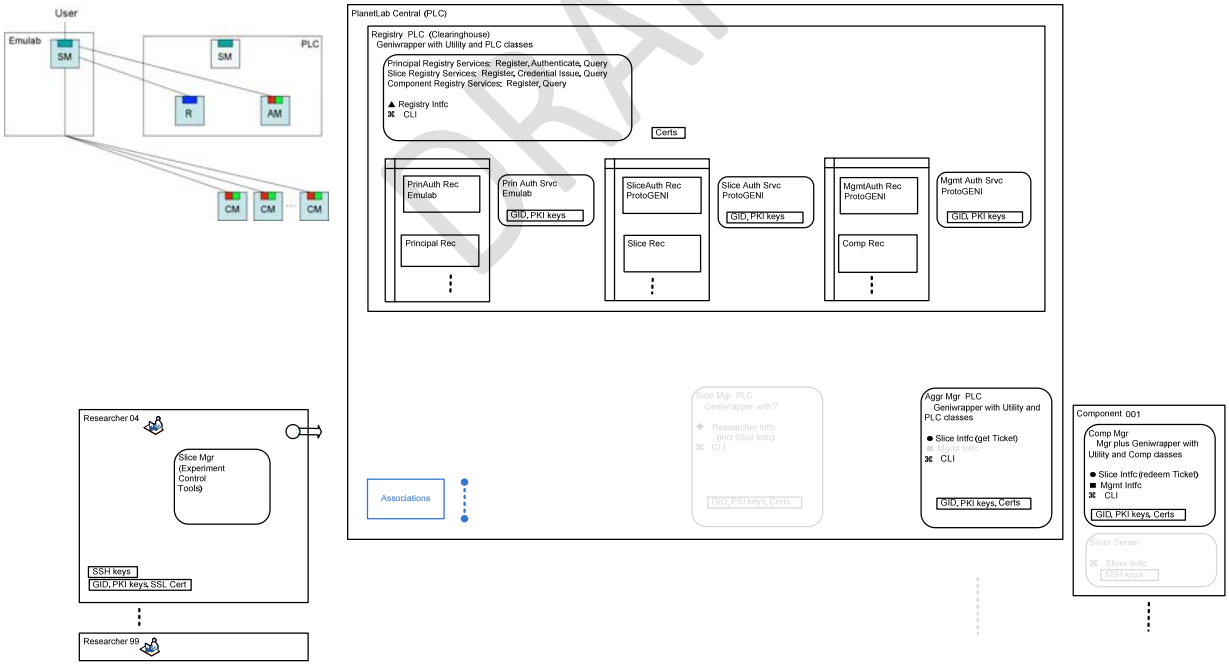

Figure 4-3. PlanetLab GENI Structure with Dedicated Slice Manager.

4.5 Slices

From a Researcher's perspective, a slice is an interconnected set of reserved resources, or slivers, on heterogeneous substrate aggregates (components). Researchers can remotely discover, reserve, configure, and program, debug, operate, manage, and teardown resources on these aggregates (components) to setup, utilize and then teardown the slivers necessary to complete an experiment. See Section 2.3.

From an Operator's perspective, slices are the primary abstraction for accounting and accountability—resources are acquired and consumed by slices, and external program behavior is traceable to a slice, respectively.

In the PlanetLab GENI suite, slices are registered in the PlanetLab database contained within PLC, and (per the SFA) principals (users) are given privileges associated with a slice, such as PI or Researcher.

4.6 Message Flows

The current approach to message flows in the PlanetLab GENI control framework is summarized by:

- Principals that periodically connect and communicate with Registries, Aggregates, Components and Slice Mangers via defined interfaces and APIs.
- In particular, Researchers periodically connect and communicate with Registries, Aggregates, Components and Slice Mangers so that they can acquire the resources necessary for them to setup and execute experiments. In these transactions, Aggregates supply resources, and Researchers consume resources.
- Since the Aggregates are expected to be widely distributed, and connections are made over an IP network that may be the open Internet, the message flows must be secure, and the Principals must be properly authenticated.

DRAFT

In the basic PlanetLab GENI suite shown in Figure 4.1, users interact with the Slice Manager (using either a GUI or a programmatic interface) to create and control their slices. The Slice Manager contacts the registry to retrieve the necessary credentials, and then invokes the Slice Interface on the aggregate to create and control the slice. As is the common case in PlanetLab, the aggregate (rather than end users) interacts with the individual nodes. Note that the current implementation of PLC uses a private interface to interact with the individual components (although the components also export the slice interface to other clients).

In the PlanetLab GENI suite utilizing an alternative slice manager as shown in Figure 4-3, the Slice Manager contacts the PlanetLab Registry to retrieve the necessary credentials. It then contacts the PlanetLab Aggregate Manager to retrieve a ticket for each slice it wants to instantiate. The alternate Slice Manager then directly contacts the PlanetLab nodes to redeem these tickets, and later, to control the slices on those nodes. Because each node only caches slice-related state, the alternate slice manager is responsible for ensuring that the slices it instantiates persist across node failures.

4.7 Registry, Slice and Management Interfaces

In PlanetLab GENI, principals periodically connect and communicate with Registries, Aggregates, Components and Slice Managers via Registry, Slice and Management Interfaces. Each uses a custom API as defined in the SFA on top of a common "GENI" protocol.

Note that PlanetLab supports an extensive Management interface that goes well beyond anything defined by the SFA. This is a private interface known only to PlanetLab operators. One can view the SFA management interface as a small subset of this PlanetLab-specific O&M interface that is ommon to all components participating in a federated slice-based facility.

The GENI protocol is based on XML-RPC. It is implemented primarily in the geniserver.py and geniclient.py files located with the utility classes in the geniwrapper software module. Modifications to the XML-RPC protocol include the following:

The transport mechanism uses HTTPS instead of HTTP.

HTTPS certificate verification is disabled so that custom GENI verification based on GID can be done instead.

When an exception occurs on the server, verbose exception information is sent to the client, to assist debugging efforts

Authentication of the client by the server is done by using Credentials/GIDs; see Section 4.12.

DRAFT

4.8 Sliver Interfaces

Once a Sliver has been created on a component, a Sliver Interface is sometimes provided on the component to allow the Researcher to program, configure and operate a server within the underlying component, here designated the Sliver Server. The Sliver Server may be a physical server, or a virtual machine. The Sliver Server may represent a component (host) itself, or a part of the component, e.g., a controller acting as a protocol engine.

Typically, a Researcher on their server connects with the Sliver Server via the Sliver Interface using a Secure Shell (SSH) login. An SSH login provides for almost complete control of the Sliver Server within the component, and it is important that that server by well isolated from other slices to prevent security breaches.

Mutual authentication is required in an SSH login. PlanetLab GENI utilizes public and private key pairs, where the public key is loaded into the Sliver Server, and both public and private keys are held in the SSH client.

Question: How will this function provided in PlanetLab GENI?

4.9 Public Key Infrastructure (PKI) and Certificates

In PlanetLab GENI, per the SFA, a Public Key Infrastructure and x509 certificates are utilized to cryptographically sign information, which can then be transferred and verified. One PKI covers all PlanetLab GENI principals and entities. Per the SFA, the certificates are signed by various authorities located within the Registry. In some cases, certificates are signed by a chain of authorities.

Geniwrapper uses two crypto libraries: pyOpenSSL and M2Crypto to implement the necessary crypto functionality. Ideally just one of these libraries would be used, but unfortunately each of these libraries is independently lacking. The pyOpenSSL library is missing many necessary functions, and the M2Crypto library has crashed inside of some of the functions. The design decision is to use pyOpenSSL whenever possible as it seems more stable, and only use M2Crypto for those functions that are not possible in pyOpenSSL.

Public-private key pairs are implemented by the Keypair class. A Keypair object may represent both a public and private key pair, or it may represent only a public key (this usage is consistent with OpenSSL).

The certificate class implements a general purpose X509 certificate, making use of the appropriate pyOpenSSL or M2Crypto abstractions. It also adds several addition features, such as the ability to maintain a chain of parent certificates, and storage of application-specific data.

Certificates include the ability to maintain a chain of parents. Each certificate includes a pointer to it's parent certificate. When loaded from a file or a string, the parent chain will be automatically loaded. When saving a certificate to a file or a string, the caller can choose whether to save the parent certificates as well.

Example creation of a certificate:

```
# create a key for an issuer
        issuerKey = Keypair(create=True)
        issuerSubject = "testissuer"
# create a key for the certificate
        userKey = KeyPair?(create=True)
# create the certificate, set the issuer, and sign it
        cert = Certificate(subject="test")
        cert.set_issuer(issuerKey, issuerSubject)
        cert.set_pubkey(userKey)
        cert.sign()
```

Verification examines a chain of certificates to ensure that each parent signs the child, and that some certificate in the chain is signed by a trusted certificate. Verification is a basic recursion:

```
if this_certificate was signed by trusted_certs:
    return
else
    return verify_chain(parent, trusted_certs)
```

At each recursion, the parent is tested to ensure that it did sign the child. If a parent did not sign a child, then an exception is thrown. If the bottom of the recursion is reached and the certificate does not match a trusted root, then an exception is thrown.

DRAFT

4.10 Global Identifiers (GIDs)

In PlanetLab GENI, per the SFA, each Principal and object is designated by a Global Identifier (GID). A GID includes a tuple of the following fields:

(uuid, hrn, public_key)

UUID is a unique identifier and is created by the python uuid module (or the utility function create_uuid() in gid.py).

HRN is a human readable name. It is a dotted form similar to a backward domain name. For example, planetlab.us.arizona.bakers.

PUBLIC_KEY is the public key of the principal identified by the UUID/HRN. It is a Keypair object as defined in the cert.py module.

It is expected that there is a one-to-one pairing between UUIDs and HRN, but it is uncertain how this would be inforced or if it needs to be enforced.

GIDs are a derivative class of certificates and as such the GID class inherits all the methods of the certificate class. The 5 fields of the GID tuple are stored in the subject-alt-name field of the X509 certificate. Two routines are included to package and unpackage these fields: Encode() and Decode(). Encode should be called prior to signing the GID. Decode is automatically called on demand by the various get_*() functions.

Verification first performs the checks of the certificate class (verifying that each parent signs the child, etc). In addition, GIDs also confirm that the parent's HRN is a prefix of the child's HRN. Verifying these prefixes prevents a rogue authority from signing a GID for a principal that is not a member of that authority. For example, planetlab.us.arizona cannot sign a GID for planetlab.us.princeton.foo.

DRAFT

4.11 Credentials

In PlanetLab GENI, per the SFA, a Researcher requires a Credential to access an Aggregate or a Component to have resources authorized and assigned to create slivers, and to control slivers.

Credentials are a derivative class of certificates and as such the credential class inherits all the methods of the certificate class. A credential is a tuple:

(GIDCaller, GIDObject, LifeTime?, Privileges, Delegate)

where

GIDCaller identifies the holder of the credential. When a credential is presented to a component, the security layer ensures that the client matches the public key that is contained in GIDCaller.

GIDObject identifies the object of the credential. This object depends upon the type of the credential. For example, the credential for a user likely has GIDObject == GIDCaller. Credentials for slices would include the GID of the slice in the GIDObject field. Credentials for authorities include the GID of the authority in the GIDObject field.

LifeTime? is the lifetime of the credential. Currently not implemented; expect to implement it as an expiration date, and refuse credentials beyond that date.

Privileges is a Rights object that describes the rights that are granted to the holder of the credential.

Delegate is a True/False bit that indicates whether or not a credential can be delegated to a different caller.

The 5 fields of the credential tuple are stored in the subject-alt-name field of the X509 certificate. Two routines are included to package and unpackage these fields: Encode() and Decode(). Encode should be called prior to signing the ticket. Decode is automatically called on demand by the various get_*() functions.

4.12 Authentication

In the PlanetLab GENI suite, authentication of a Principal's client is done by the server at a Registry, Slice or Management Interface.

Both the client and server specify their private and public keys when opening an SSL socket, and upon successful connection establishment, each knows the other's public key (by convention, this key is stored in an X.509 certificate).

Generally, each operation contains a credential as the first argument. This credential includes the GID of the caller, which in turn contains the public key of the caller. The server ensures that this public key matches the public key that is being used to decrypt the HTTPS connection, thus ensuring the caller must possess the private key that corresponds to the GID.

Next, the geniwrapper module decode_authentication routine is used to verify that the credential gives the caller the right to invoke the corresponding operation and that the credential is properly signed (recursively) by its parents.

Authentication of the server by the client is left as an exercise for the client. It may be done easily by specifying the server's public key when the client create the HTTPS connection. This presumes the client knows the public key (or GID) of the server he is trying to connect to.

DRAFT

4.13 Authorization

Authorization in the PlanetLab GENI suite, per the SFA, is mediated by Credentials and Tickets which are presented by Researchers to Registries, Aggregates and Components.

For example:

A Researcher (using a Slice Manager) presents a Slice Credential to an Aggregate Manager to retrieve (get) a Ticket that authorizes the Researcher to use certain specified resources.

Next, the Researcher presents the Ticket to a Component Manager to instantiate the slice and have the authorized resources actually assigned.

Finally, the Researcher presents the Slice Credential to the Component Manager to control the slice (sliver).

4.14 Tickets

In PlanetLab GENI, per the SFA, a Ticket is a signed certificate that authorizes the holder to use certain specified resources. Similar to GIDs and Credentials, tickets also leverage the certificate object.

A Ticket is a tuple:

(gidCaller, gidObject, attributes, rspec, delegate)

where

gidCaller = GID of the caller performing the operation

gidObject = GID of the slice

attributes = slice attributes (keys, vref, instantiation, etc)

rspec = resources

Tickets are created by invoking GetTicket?() on the plc wrapper. The slice attributes and rspec are taken from the planetlab slice database and represent the current state of the slice. As of yet, tickets do not include any concept of time -- a ticket represents the state of the slice at the current time only.

Tickets are redeemed by invoking RedeemTicket?() on the slice interface. The attributes and spec are combined back into a planetlab slice record and handed off to the node manager.

Tickets are signed by an authority and include parentage information that traces the chain of authorities back to a trusted root.

Verification of a ticket uses the standard parentage verification provided by the certificate class. Specifically, each certificate is signed by a parent, and some certificate must resolve to the trusted root set that is specified on the component.

Unlike credentials and GIDs, the parent of a ticket may be a degenerate ticket that does not include the full 5-tuple (caller, object, attributes, rspec, delegate). In such a case, the parent is just a placeholder in the chain of authority used to convey the parentage information.

Delegation of tickets is not something that is discussed in the SFA, but it is supported in the ticket class and may be a useful feature. For example, Alice may hold a ticket for a particular component, and delegate that ticket to Bob. Bob could then instantiate a slice for Alice. This may be one way to implement a slice manager.

DRAFT

4.15 Resource Specification (RSpec)

A *resource specification* (RSpec) describes a component in terms of the resources it possesses and constraints and dependencies on the allocation of those resources.

In PlanetLab GENI, the rspec is currently a dictionary of {name: value} pairs. These pairs are taken verbatim from the planetlab slice database.

The general rule that is used is that things in the slice record that do not specifically imply a tangible resource (initscripts, keys, etc) are treated as attributes and things that do specify a tangible resource (disk, network, etc) are treated as the rspec.

The definition of an rspec is evolving. It remains to reconcile the eclipse schema with Geniwrapper; see http://svn.planet-lab.org/wiki/PLDataModel . Gacks is also using another rspec format, which may be need to be reconciled with the eclipse schema and/or geniwrapper.

5 Principals in the PlanetLab GENI Control Framework

5.1 Identification

Each principal (user) in PlanetLab GENI, such as a Researcher, has a unique Global Identifier (GID) that does not change, and which includes a Human Readable Name (HRN), a UUID and a Public Key; see Section 4.10.

Question: Who creates the GID, including UUID and HRN?

5.2 Registration

Each principal (user) is registered in the Principal Registry, part of the PLC Registry, implemented on top of the PLC database; see Section 4.1.

Once registered, a Principal can be authenticated, and can be identified so that they can be given a appropriate privileges.

Question: Who does this? How is it done?

5.3 Authentication

Each PlanetLab GENI principal (e.g., Researcher) can be authenticated by the server at a Registry, Slice or Management Interface. Generally, each operation contains a credential as the first argument. This credential includes the GID of the caller, which in turn contains the public key of the caller. The server ensures that the public key from the credential matches the public key that is being used to decrypt the HTTPS connection; see Section 4.12.

DRAFT

6 Aggregates and Components in PlanetLab GENI Control Framework

6.1 Identification

Each Aggregate and Component in PlanetLab GENI has a unique Global Identifier (GID) that does not change, and which includes a Human Readable Name (HRN), a UUID and a Public Key; see Section 4.10.

Question: Who creates the GID, including UUID and HRN?

6.2 Registration

Each Aggregate and Component must be registered in the Component Registry, part of the PLC Registry, implemented on top of the PLC database; see Section 4.1.

Question: Who does this? How is it done? What information is included?

6.3 Resource Allocation

Components (nodes) implicitly delegate control over their resources to PLC (the aggregate), which is responsible for implementing PlanetLab's resource allocation policy.

Question: How is it done? How does the Aggregate Manager know what Tickets can be issued? How is this based on entries in the Component Registry?

DRAFT

7 Slices in PlanetLab GENI Control Framework

7.1 Identification

Each Slice in PlanetLab GENI has a unique Global Identifier (GID) that does not change, and which includes a Human Readable Name (HRN), a UUID and a Public Key; see Section 4.10.

Question: Who creates the GID, including UUID and HRN?

7.2 Registration

In the PlanetLab GENI suite, slices are registered in the PlanetLab database contained within PLC, and (per the SFA) principals (users) are given privileges associated with a slice, such as PI or Researcher.

Question: Who does this? How is it done? What information is included?

7.3 Credential Issue

In PlanetLab GENI, a Researcher (or another principal with appropriate privileges) requests a Slice Credential from the Slice Registry in the PLC Registry.

The Slice Credential is signed by the Slice Authority and can then be used by the researcher to retrieve (get) a Ticket, etc.

Question: But, the Researcher needs to present a credential to the PLC Registry when it requests a Slice Credential. How can this process be bootstrapped?

Consider this process:

- A Principal assembles their basic identity (GID, including GlobalName and optional ObjectID) and authentication (PrivateKey and PublicKey pairs) records, and the administrator of one of their responsible principal authorities creates a Principal Record for them, and registers it in the Principal Registry.
- The Principal calls the Principal Registry Interface with a GetCredential request where both the calling and referenced objects are themselves, identified with their GID.
- The registry matches the GID with the existing Principal Record, and verifies their identity via the certificate carried over the SSL connection.
- Finally, the registry issues the Principal a “self credential”, which can then be used to access a registry to get other credentials.

Now that a Principal has a “self credential” identifying themselves, they can use it to request additional credentials at other registries. These other registries must recognize the signatures on the “self credential”.

For example, following the SFA, a Researcher can use their “self credential” to call the Slice Registry Interface with a GetCredential request, with the referenced object being a slice to get a “slice credential” that will allow them to create slivers, etc.

Question: Is there only one Slice Credential that the Researcher holds and uses repeatedly, or must they repeatedly get credentials, that then have only “one use”?

DRAFT

8 Experiment Setup in PlanetLab GENI Control Framework

The PlanetLab GENI control framework provides all of the basic functions necessary for a GENI researcher to setup an experiment.

8.1 Resource and Topology Discovery

The PlanetLab GENI control framework allows a Researcher using a Slice Manager to discover all of the resources available to them from the Aggregate(s) associated with the PlanetLab GENI suite.

This is done by contacting the Component Registry in the PLC Registry, calling LISTCOMPONENTS, and compiling the results in the Slice Manager.

Question: How exactly is this done? What information is returned?

Question: How can they discover their interconnection topology?

8.2 Resource Sharing

In PlanetLab GENI, each component provides for resource sharing among multiple researchers, referencing multiple slices, and assigns each researcher their own sliver or slivers.

DRAFT

8.3 Resource Authorization and Policy Implementation

The PlanetLab GENI control framework allows an Aggregate to authorize the assignment of resources to a Researcher referencing a particular Slice, following local policies. See Figure 8-1.

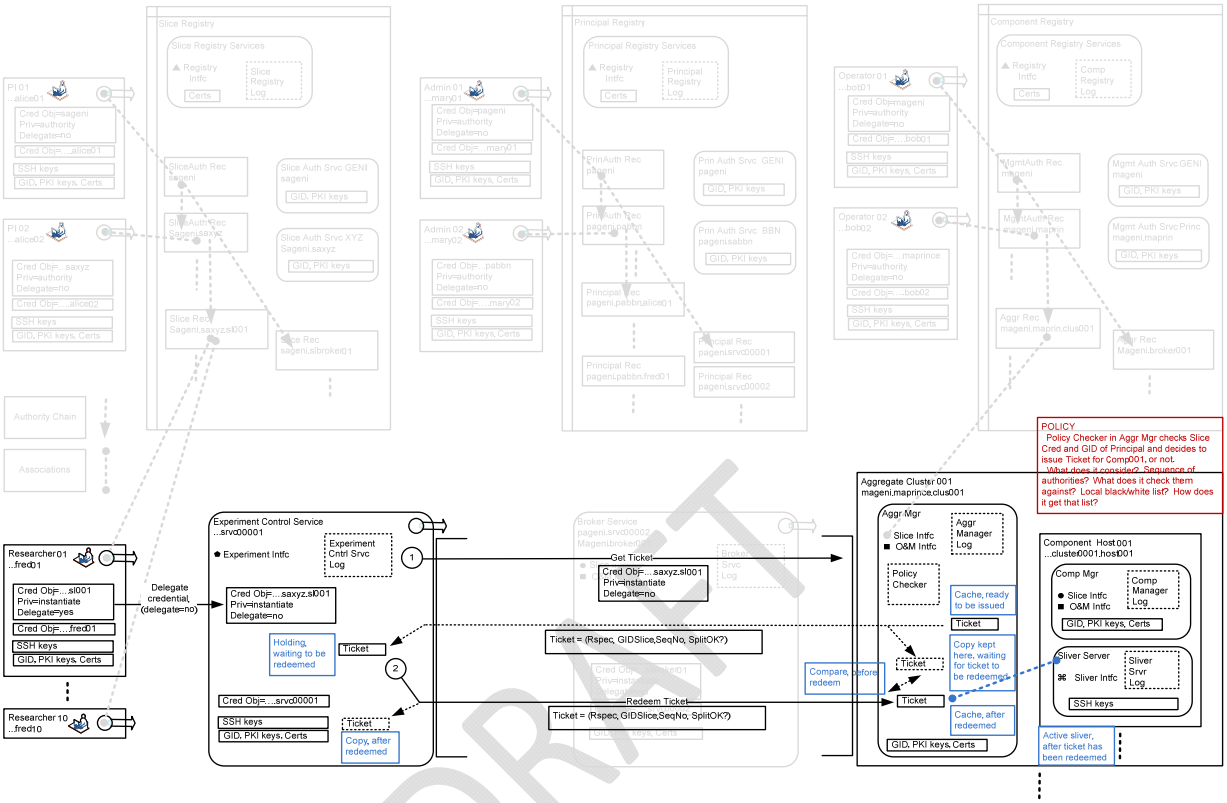

Figure 8-1. Resource Authorization plus Assignment

Step 1a: To URL of Slice Interface on Aggregate Manager, Researcher using Slice Manager presents Slice Credential and and desired RSpec, and calls GETTICKET.

Step 1b: Aggregate decides using its local policy that it will authorize resources to this Researcher and the referenced Slice on a "best effort" basis, and then returns a Ticket to the Researcher.

Question: What is local policy? Does the Aggregate give a Ticket to anyone with a valid Slice Credential, i.e., there is an "always yes" local policy?

8.4 Resource Assignment

The PlanetLab GENI control framework allows a Component to assign resources to a Researcher presenting a valid Ticket. See Figure 8-1.

Step 2a: To URL of Slice Interface on Component Manager, Researcher using Slice Manager presents Ticket and calls REDEEMTICKET.

Step 2b: Component assigns resources on a "best effort" basis to the Researcher and their Slice, and a Sliver is created.

Question: Is there a way to set starting time and duration?

Question: Is there a way to get a firm commitment?

8.5 Component Programming

The PlanetLab GENI control framework should allow a Researcher to login to an assigned sliver (Component), load code, and then boot it, etc.

This could be done via a Sliver Interface; see Section 4.8.

Question: How will this function provided in PlanetLab GENI?

The PlanetLab GENI control framework allows a Researcher (or Operator) to reboot an assigned sliver (Component) using the Management Interface on the Component. See Figure 8-2.

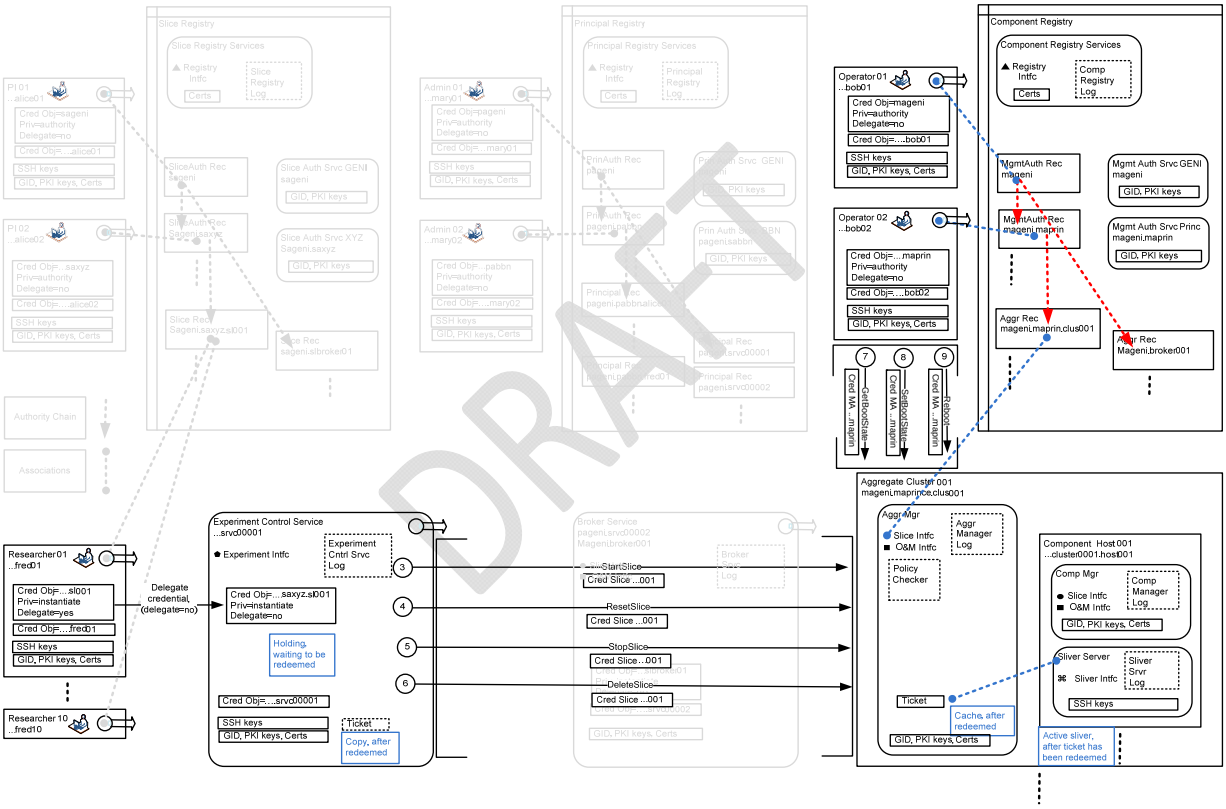

Figure 8-2. Managing a Component

The management interface includes three operations:

SetBootState(Credential, State)

State = GetBootState(Credential)

Reboot(Credential)

The first operation is used to set the boot state of a component to one of the following four values: debug (component fails to boot, but should keep trying), failure (component is experiencing hardware failure, and so is taken offline until a human intervenes), safe (component available only for operator diagnostics), or production (component available for hosting slices). The second operation is used to

learn a component's boot state and the third operation forces the component to reboot into the current boot state.

Note that we expect a given Component to support a much richer set of management-related (O&M) operations, effectively extending the required operations listed here. The management interface defines only the minimal set of operations **all** components must support.

8.6 Disconnected Operation of Components

In a GENI suite, some of the components (such as wireless servers) will require "disconnected operation", where they are controlled and polled in the short periods of time that they are connected to the suite.

(Note yet defined in PlanetLab GENI.)

8.7 Resource to Resource Connections

When a researcher has been assigned resources from two (or more) aggregates that must be connected together, the PlanetLab GENI control framework provides a way for the researcher to learn about the connection points, request the connections, following the necessary sequence, and receive a verification that the connection has been completed.

(Note yet defined in PlanetLab GENI.)

8.8 Setup Verification

When a researcher has been assigned resources on GENI (or federated) aggregates for an experiment, the control framework should provide a way for the researcher to ask the aggregates to verify the setup before it is time for the experiment to start.

(Note yet defined in PlanetLab GENI.)

DRAFT

9 Experiment Execution in PlanetLab GENI Control Framework

9.1 Experiment Control

When a Researcher, associated with a designated slice, has been assigned resources on aggregates for an experiment, the PlanetLab GENI control framework provides a way for a Researcher to control the slices/slivers in the components using commands appropriate to the nature of the sliver. See Figure 8-2.

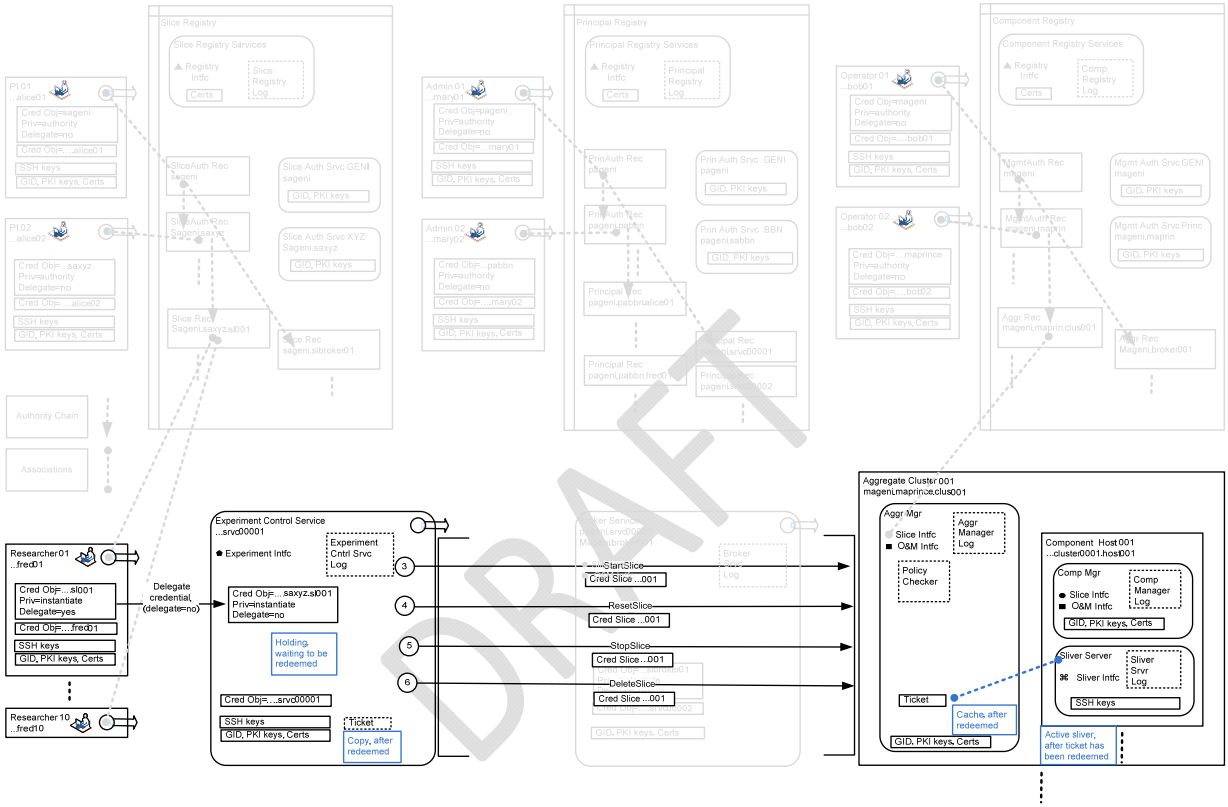

Figure 9-1. Controlling a Slice/Sliver.

Per the SFA, here are four general commands, made by a Researcher using a Slice Manager and presenting a SliceCredential:

Command 1: StartSlice

Command 2: ResetSlice

Command 3: ResetSlice

Command 4: DelteSlice

An alternate approach would be for the researcher to use a sliver credentials. A sliver credential would be identical to a slice credential, but would 1) only be redeemable on a particular component, and 2) would resolve to a trusted_root unique to that component (likely the component's GID

certificate). Sliver credentials would be returned by the RedeemTicket? call and would give the caller the permission required to start and stop the sliver, etc.

Sliver credentials are not yet implemented in geniwrapper, but their implementation would be straightforward.

9.2 Experiment Data Collection and Management

GENI will provide for experiment data collection and measurement, both locally within aggregates (components) and globally in designated measurement services. It is expected that large data files will be gathered by these services, and that they will need to be transferred to a software repository and/or an experiment analysis service after an experiment.

To accomplish this, the control framework should provide the mechanism(s) to allow a researcher to transfer large software records between components, software repositories, etc. For example, the control framework could provide a file transfer service based on ftp.

(Not yet defined in PlanetLab GENI).

9.3 Forensic and Usage Data Collection and Management

Forensic and usage data from a GENI suite has many uses, including:

- Finding anomalies that indicate errors, faults, malicious activity, etc.
- Allowing help desk functions to be provided to researchers.
- Permitting proper administration and management of suite resources.
- Permitting financial accounting where necessary.

The control framework should provide a structure for collecting and managing forensic and usage data, including formats and log structures.

(Not yet defined in PlanetLab GENI.)

9.4 Experiment Status Monitoring

Experiment status can be monitored in a GENI suite by:

- Defining trigger events associated with the use of resources in a sliver, an aggregate or a component.
- Defining watchdog processes, to periodically verify functions in a sliver, an aggregate or a component.
- Sending notifications to a researcher, an administrator, an operator, or anyone who has requested receipt.

The control framework should provide a structure for experiment status monitoring, and sending notifications.

(Not yet defined in PlanetLab GENI.)

DRAFT

9.5 Experiment Status Commands

The control framework must provide a structure for commanding changes in the status of resources used by an experiment. It must be possible for changes to be commanded by a researcher or by an aggregate or component administrator or operator. For example, it must be possible to command "shutdown all slivers in this aggregate associated with slice x".

(Not yet defined in PlanetLab GENI.)

DRAFT

10 Federation in PlanetLab GENI Control Framework

10.1 Federated Aggregates and Components

A PlanetLab GENI control framework should provide for the inclusion of a variety of federated aggregates (and their included components) to provide a wide range of resources to the Researchers. One possible approach is shown in Figure 10-1, where an Aggregate Manager for VINI resources is included in the PlanetLab GENI suite.

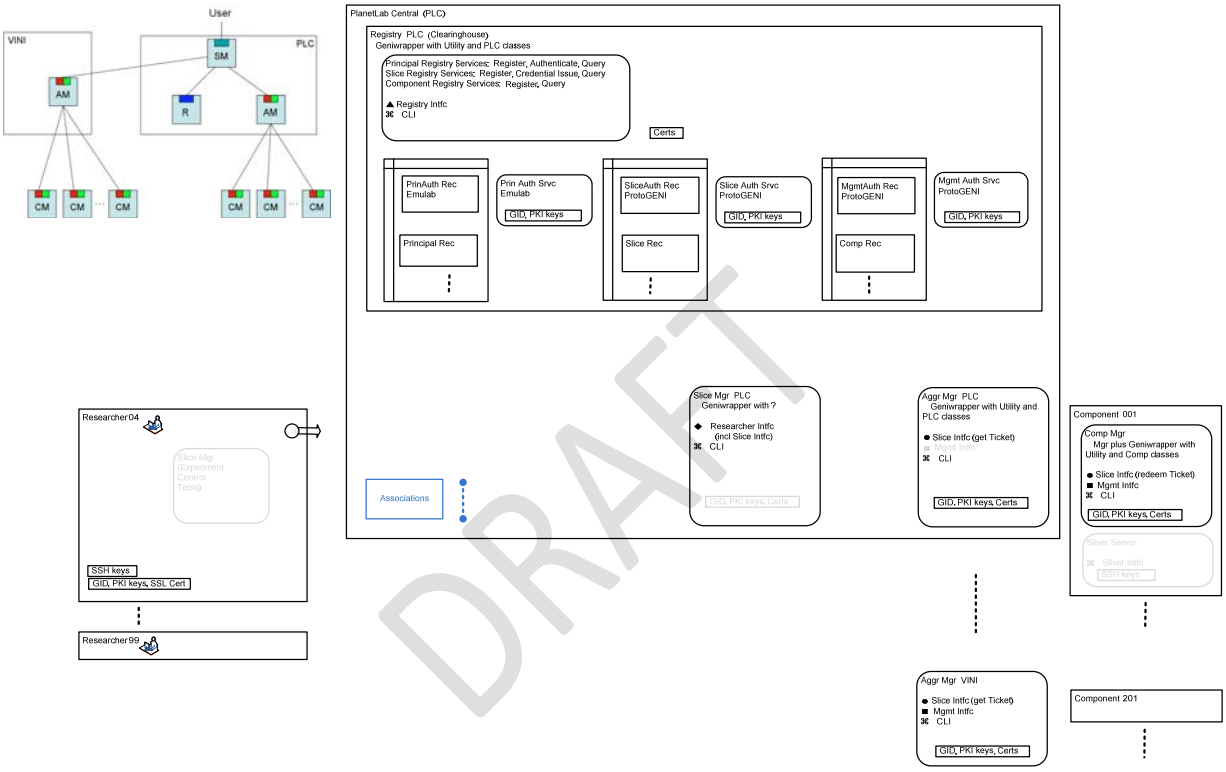

Figure 10-1. Federated Aggregates and Components

This scenario spans multiple aggregates—PlanetLab and VINI—each responsible for its own set of components. That is, VINI and PlanetLab are distinct management authorities, each responsible for a distinct aggregate of components. In this case, VINI does not operate its own registry or slice manager, and PlanetLab's Slice Manger presents users with a unified view of all the components available on both systems, hiding the fact that its global view spans multiple aggregates.

To create a slice, the PlanetLab SM would need to contact both available aggregate managers to learn about the available components. It would then present these components to the user in an SM-specific way. Once the user selects the set of components to be included in his or her slice, the SM would call the Registry to retrieve the necessary credentials, and then invoke the InstantiateSlice operation on the respective aggregates to create the cross-aggregate slice.

Note that in this scenario (and the one described in the next section), the SM plays the role of an *aggregate of aggregates*. When viewed from this perspective, it makes sense that the SM exports the slice interface, just like any other aggregate (i.e., the Researcher Interface is a superset of the slice interface).

DRAFT

10.2 Federated Suites

The control framework should provide for the federation of a GENI suite with other suites, where each suite has its own complete set of entities, but is independently owned and operated.

One possible approach is shown in Figure 10-2, where a PlanetLab Europe GENI suite is federated with the PlanetLab Central GENI suite.

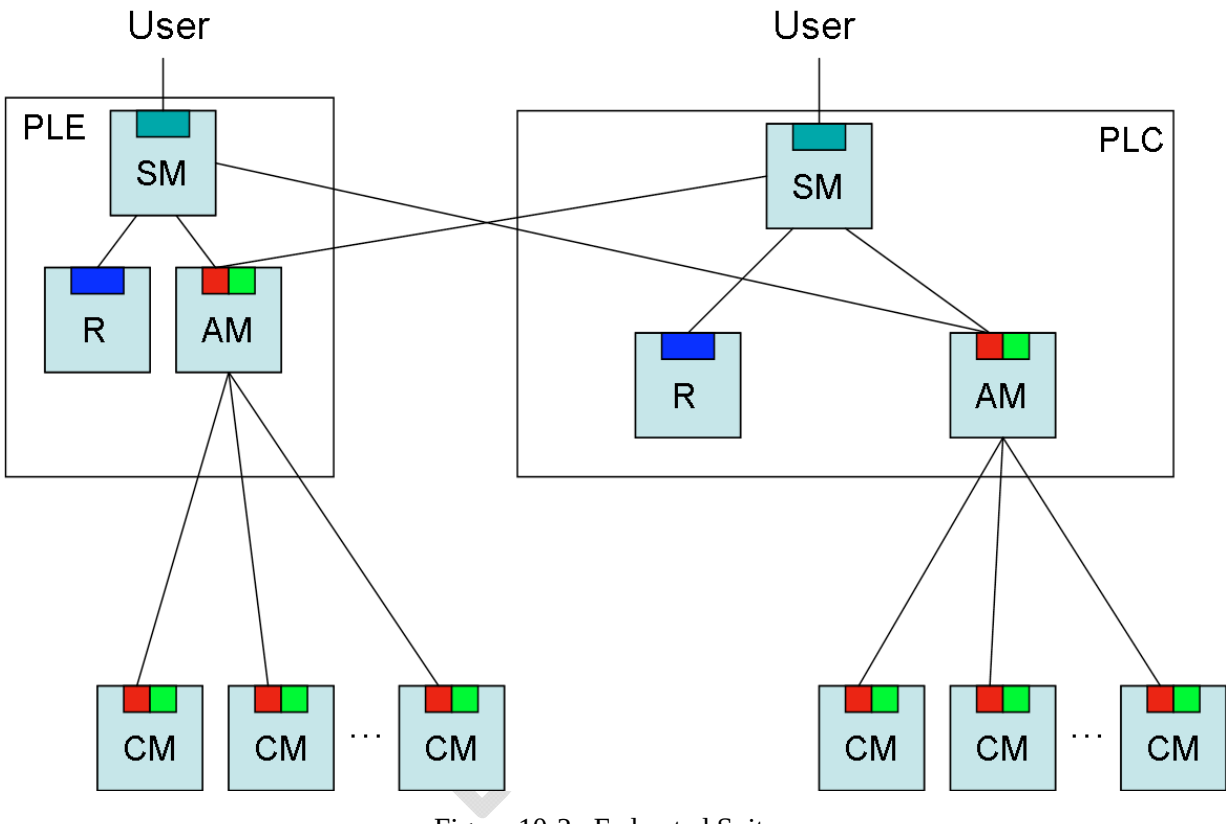

Figure 10-2. Federated Suites.

Our final scenario, shown in Figure 3.6, involves symmetric federation between two autonomous aggregates, one representing PlanetLab Europe (PLE) and the other representing the rest of PlanetLab (PLC). Both systems support their own slice manager, registry servers, aggregate manager, and set of components.

As in the previous scenario, users interact with their “local” SM, which creates and manages slices spanning both aggregates.

Although not explicitly depicted in the figure, the PLC registry points to the PLE registry. That is, registry records for the top-level PlanetLab authority, including the record for the EU subauthority, are maintained in the PLC registry, while records associated with the EU subauthority are maintained in the PLE registry server.

11 PlanetLab GENI Cluster B Spiral 1 Implementation

11.1 Start of Spiral 1

At the beginning of Spiral 1, the PlanetLab GENI Cluster B implementation includes the PLC Registry, the PLC-based Aggregate Manager, associated PLC components, other associated component and a PLC-based Slice Manager. See Figure 11-1.

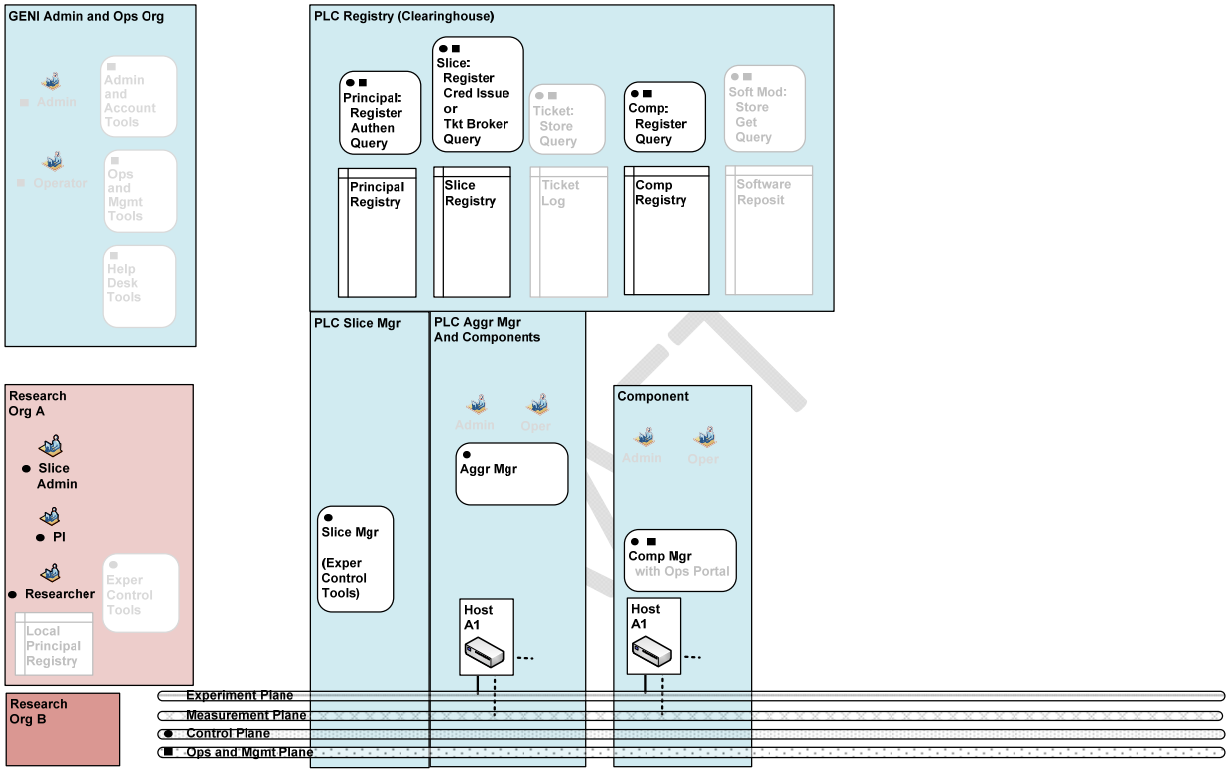

Figure 11-1. Start of PlanetLab GENI Cluster B Spiral 1 Implementation.

Question: What components from other Cluster B projects are included?

Question: Are there any other aggregates?

Question: When are additional Slice Managers introduced?

11.2 Completion of Spiral 1

Eventually, the following Cluster B projects will be integrated into the PlanetLab GENI implementation; see Figure 11-2.

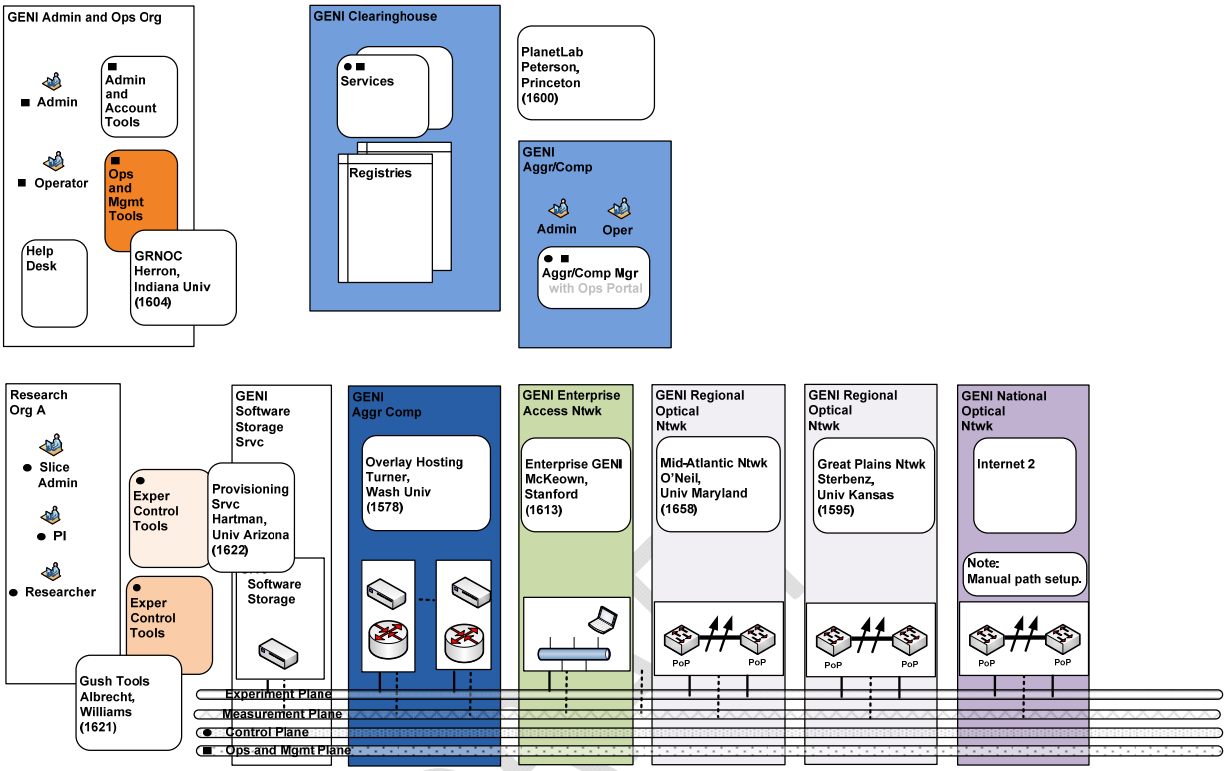

Figure 11-2. PlanetLab GENI Cluster B Implementation.

DRAFT