

# **GENI Control Framework Structure**

## **Implementations**

January, 2011

Darwin Witt

## TABLE OF CONTENTS

1	DOCUMENT SCOPE
	1.1 Purpose of Document
	1.2 Related Documents
1.2.1	GENI Documents
1.2.2	Standards Documents
2	CONTROL FRAMEWORK STRUCTURES
	2.1 Overview of ProtoGENI Control Framework Structure
	2.2 Overview of PlanetLab Control Framework Structure
3	REGISTRIES
	3.1 ProtoGENI Registries
	3.2 PlanetLab Registries
4	AGGREGATES AND COMPONENTS
5	PRINCIPALS
6	SERVICES
7	SLICES
8	MESSAGE FLOWS
	8.1 Message flows in ProtoGENI
	8.2 Message flows in PlanetLab
9	GOALS FOR FEDERATION
10	DEFINITIONS

## **1 Document Scope**

This section describes this document's purpose and the set of related documents.

### **1.1 Purpose of this Document**

This document provides an overview of current implementations of the specified GENI Control Framework Structure defined by the GENI Project Office. The central goal of this overview is to determine the differences in implementations and identify goals for moving forward with establishing a fully-federated control framework.

### **1.2 Related Documents**

The following is a listing of all documents used directly as resources for the compilation of this document.

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

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

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

Some of the material in this document is taken from a draft "Slice-Based Facility Architecture (SFA)" document, Draft v2.0, July 2010, by Larry Peterson, Robert Ricci, Aaron Falk, and Jeff Chase.

#### **1.2.1 GENI Documents**

GENI Spiral 2 Overview - June 3, 2010 - Document ID:GENI-INF-PRO-S2-OV1.1

GENI Control Framework Requirements - January 9, 2009 - Document ID: GENI-SE-CF-RQ-01.3

ProtoGENI Control Framework Overview - February 27, 2009 - Document ID: GENI-SE-CF-PGO-01.4

PlanetLab Control Framework Overview - January 14, 2009 - Document ID: GENI-SE-CF-PLGO-01.2

GENI Aggregate Manager API - Sept 1, 2010 - Document ID: GENI-SE-CF-AMAPI-01.0

### **1.2.1 Standards Documents**

Slice-Based Facility Architecture (SFA), Draft v2.0, July 2010, by Larry Peterson, Robert Ricci, Aaron Falk, and Jeff Chase.

## **2 Control Framework Structures**

Clearinghouse entities required: Principal Registry, Component Registry, Slice Registry, an optional ticket log for holding "sliver records," and an optional software repository for holding software objects.

Aggregate/Component entities required: Aggregate Manager, an optional Component Manager for components that are part of an aggregate, and an optional broker service, which is basically an aggregate-of-aggregates manager

Entities defined as a "principal" who is utilizing, administering, or managing experiment resources include a Principal acting from a server utilizing a browser client and/or helper tools as well as a Principal service acting on behalf of a principal.

The GENI Control Framework defines interfaces between all entities, message types including basic protocols and required functions, and message flows necessary to realize key experiment scenarios.

The SFA defines four key types of entities operating in control framework.

Owners of parts of the network substrate, who are therefore responsible for the externally

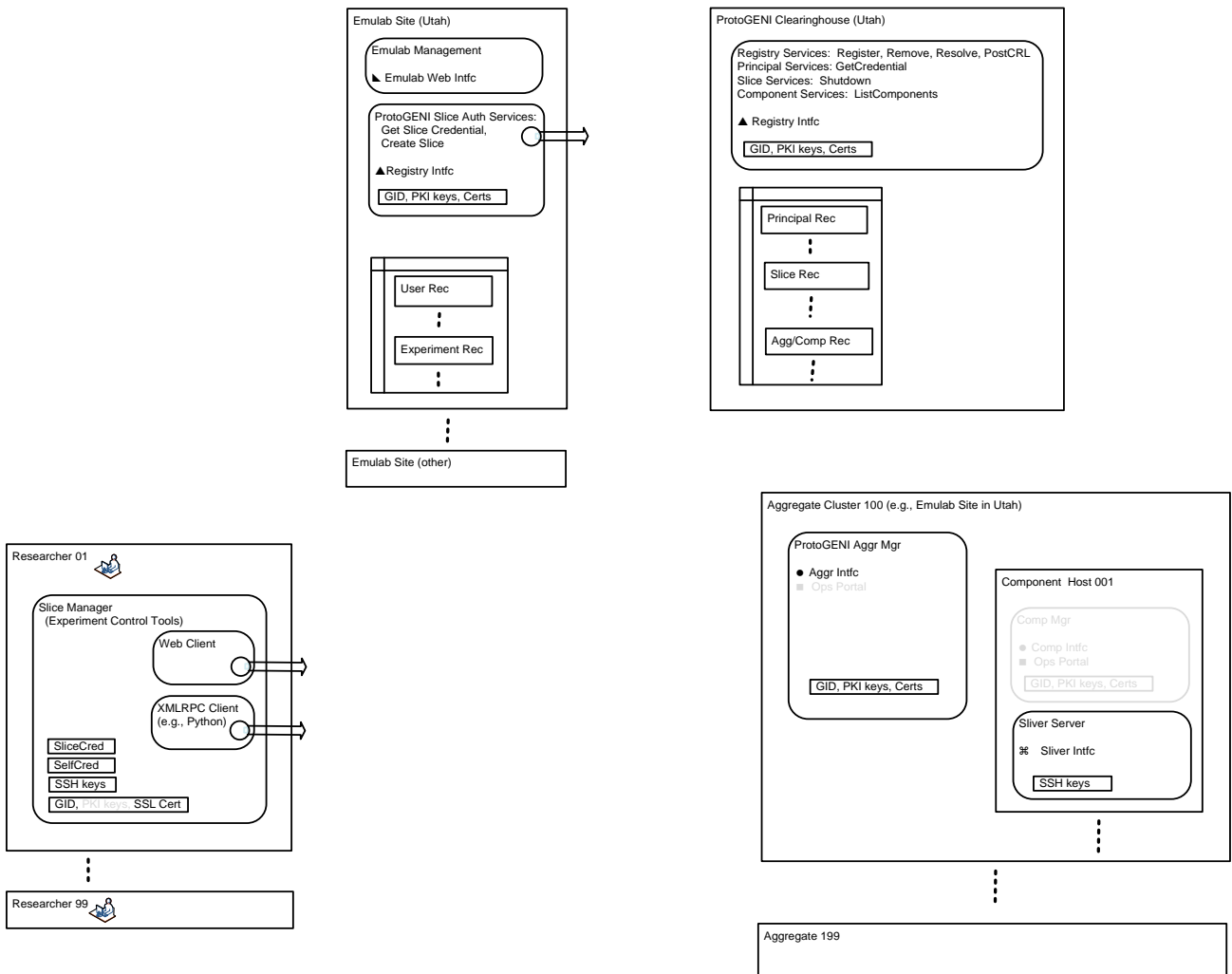
visible behavior of their equipment, and who establish the high-level policies for how their portion of the substrate is utilized.

Operators of parts of the network substrate, often working for owners, whose job it is to keep the platform running, provide a service to researchers, and prevent malicious or otherwise damaging activity exploiting the platform.

Researchers (and developers) employing the network substrate, for running experiments, deploying experimental services, measuring aspects of the platform, and so on.

Identity anchors drive authorization by asserting attributes (or roles) of other entities. These anchors are sometimes called Identity Providers or IdPs. For example, an IdP might assert that a given user is a Principal Investigator (PI) representing a research organization that can authorize individual researchers to access the facility.

## **2.1 Overview of ProtoGENI Control Framework Structure**



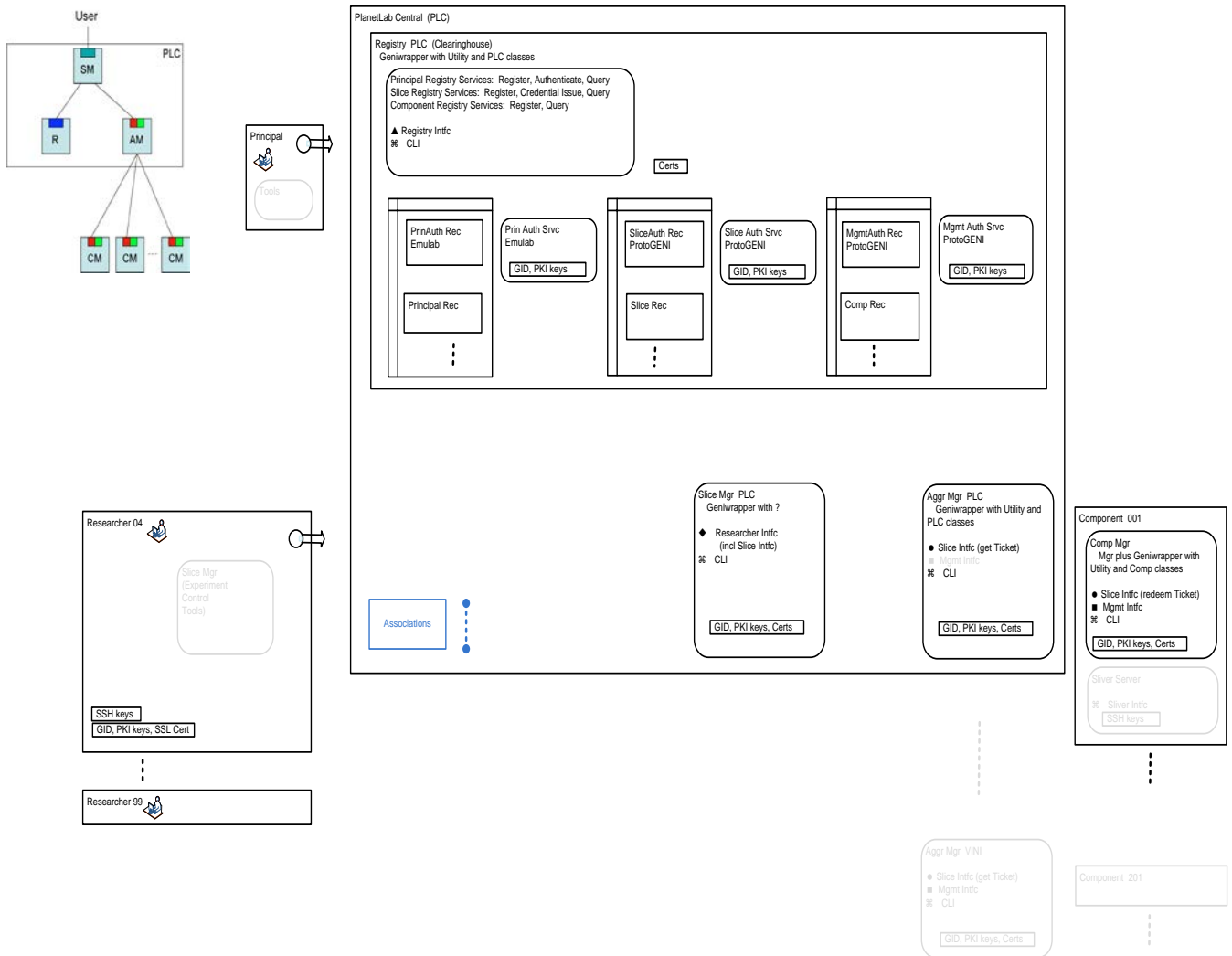
[ProtoGENI Control Framework Structure]

There are four main structures in the ProtoGENI Control Framework.

ProtoGENI is centered around a clearinghouse with the following entities: Principal Registry; Slice and Aggregate/Component Records; Common Registry Services; and Principal, Slice, and Component Services

The other three structures are the following: a Slice Authority, inherited from Emulab; Aggregates which have a ProtoGENI AM and components, each of which comprise all of the resources accessible within an Emulab Site; and Principals (users).

## 2.2 Overview of PlanetLab Control Framework Structure



[PlanetLab Control Framework Structure]

The current implementation uses PlanetLab Central and the geniwrapper software module (utility and PLC classes).

The following entities shown in the Control Framework Structure are provided by geniwrapper: PLC Registry, which realizes the Principal, Slice, and Component Registries and Services; PLC Aggregate Manager; PLC Slice Manager; and Component Managers.

The geniwrapper module is distributed as part of the MyPLC software package The geniwrapper module also defines the following sets of classes: Utility classes that implement GIDs, credentials, tickets, and an underlying secure remote invocation mechanism; PLC classes that implement the registry, slice, and management interfaces exported by AM and Registry co-located with PLC; and Component classes that implement slice and management interfaces exported by the Component Manager co-located with the node manager of each node.

The geniwrapper module also includes a command-line client program that can be used to exercise the AM, CM and Registry servers.

### **3 Registries**

#### **3.1 ProtoGENI Registries**

All registries are centrally located within the Clearinghouse, which is implemented and co-hosted with Emulab in Utah. The registry exports a Registry Interface as defined by the SFA. Each registered item has a GID that includes a UUID and a Global NAME. The UUID is a random number that is guaranteed to be unique. UUIDs of items never change.

The registry can look up an item by either GNAME or UUID.

Principals and Slices are registered in the Clearinghouse by the SA at each Emulab instance

The following tasks are performed by the Clearinghouse: registration, deletion, and resolution of various principle objects, currently slices, users, SAs, AMs, and components; listing of all known AMs, including a URL which a client can contact to get list of available resources; exports a centralized "emergency shutdown" facility can be used to terminate slices; assists in registration of new federates; collects and distributes Certificate Revocation Lists from all federates (CRLs are generated periodically by federates so that others in federation can be made aware of users that have been terminated or have had to get a new GID).

Only one instance of the Clearinghouse is in operation. The architecture eventually calls for multiple Clearinghouses, but there is no support for federating multiple clearinghouses together. Everything is stored in a MySQL database. The URL of the Clearinghouse is hardwired in scripts.

Most operations are redirected to known SAs and CMs, end users do not access the clearinghouse for most things. SAs and CMs are the only ones that can register and delete entries, and resolve records.

## 3.2 PlanetLab Registries

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

Each Record in PLC Registry is given by (Name, GID, Type, Info) where:

Name - Human Readable Name (HRN) of the object

GID - GID of the object

Type - user|sa|ma|slice|component

Info - three sub-fields:

Pointer- pointer to records in PL database

pl\_info - planetlab-specific info (when talking to client)

geni\_info - geni-specific info (when talking to client)

Pointer is interpreted depending on the type of the record. Ex: if 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, as long as records are of different types. Defined by the SFA, the basic functionality is to map HRNs into records; however, due to interactions between geniwrapper and PLC, the registry does more than act as a simple database.

Registry performs API calls on PLC that create slices, sites, users, etc. This might cause slices to be instantiated on components, because components are linked to PLC

Mapping of GENI objects to PlanetLab objects:

slice - slice

user - person

component - node

sa - site

ma - site

In a unified registry, a registry that serves as slice and component registry, the SA and MA would map to the same site record in the PLC database; however, there are two distinct GENI records for SA and MA.

Each registered object has a GID that includes a UUID and the object's Public Key.

PLC Registry includes two command-line interfaces: End-User CLI and Developer CLI.

The End-User CLI allows users to look through and update the registry as well as create and control slices. The Developer CLI is used to create and control slices, authorities, users, nodes getting and redeeming tickets

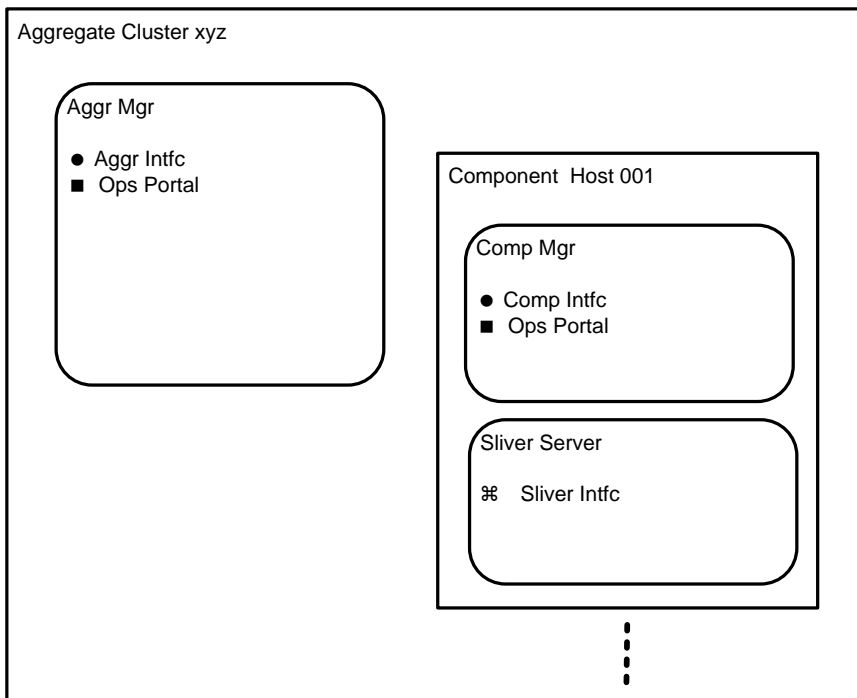
#### **4     **Aggregates and Components****

Components are grouped into aggregates, and all of the components of an aggregate are under the authority of the same MA, which also governs the aggregate. Each aggregate is controlled via an aggregate manager (AM), which exports a well-defined, remotely accessible interface. An aggregate manager with a one-component aggregate can be called a Component Manager (CM).

The AM/CM defines operations available to user-level services to manage the allocation of component resources to different users and their experiments.

A management authority establishes policies about how the aggregate's resources are assigned to users.

An aggregate may include zero, one, or many components and may optionally reveal an "internal structure" of one or more components.



[ Aggregate and its internal structure]

## 5 Principals

A GENI principal is a person acting from a server utilizing a browser client and/or helper tools who is utilizing, administering, or managing experiment resources in a GENI suite. A GENI service acting on behalf of a principal may function as a principle in a GENI suite. Each principal shall have a globally-unique name and/or numerical identifier. It should be possible to identify a principal who is acting within the GENI suite.

Principals are registered within the GENI suite, which holds a "principal record." Principals may be indirectly registered, in which case the GENI suite may recognize their registration within their home organization and check with its registration service as needed.

Principals should be registered jointly by a principal administrator who acts for the GENI suite and by one who has been authorized to act for a research organization, or their delegates, who are then jointly responsible for the registration record of that principal.

The Principal Record includes the following: Identity and contact information; Status and

quality of verifying identity; Status to operate within the GENI suite; and Information so that they can be authenticated when operating in the GENI suite (PublicKey).

Principals can serve more than one role, but not multiple registrations for multiple roles.

Typical Roles expected include the following: Administrators, Operators, Principal Administrators, Aggregate (component) administrators, Operators, Slice administrators, PIs who act for a research organization, and Researchers.

The Slice-based Facility Architecture defines the following four main principals.

A management authority (MA) is responsible for some subset of substrate components: providing operational stability for those components, ensuring the components behave according to acceptable use policies, and executing the resource allocation wishes of the component owner.

A slice authority (SA) is responsible for one or more slices. It names and registers the slices and enables users to access and control their slices. The SA must provide a contact interface to obtain information about the slice or to respond to any perceived misbehavior by the slice. MAs have the right to select which SAs are empowered to create slices on their resources.

A user is a person playing one or more roles in a facility: a researcher that wishes to run an experiment or service in a slice, an operator that manages some part of the substrate, a PI at an institution that conducts research on the facility, or an owner that contributes resources to a facility.

## **6 Services**

Services are deployed to assist in setting up and running experiments or the GENI suite in general. Currently, only the PlanetLab implementation deploys and uses services. This section describes services employed by PlanetLab.

PLC and geniwrapper are used in this context to setup Slice Managers. A prototype is currently being used which exports the slice interface and call the slice interface on the set of aggregates with which the PLC is peered.

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 may also utilize an alternative slice manager, which could be located in a centralized location, or dedicated to one Researcher.

## **7 Slices**

Three stages in the lifetime of a slice:

Register - The slice exists in name only and is bound to a set of users

Instantiate - The slice is instantiated on a set of components and resources assigned to it.

Activate - The slice is activated, at which point it runs code on behalf of a user.

A slice must be registered and bound to a user before instantiation and must be instantiated before it can run code. Slices are registered in the context of a slice authority - a principal that takes responsibility for the behavior of a slice. A slice is registered only once and has finite lifetime, the slice authority must refresh registration. Instantiating a slice configures the slice on a set of components -this can happen multiple times.

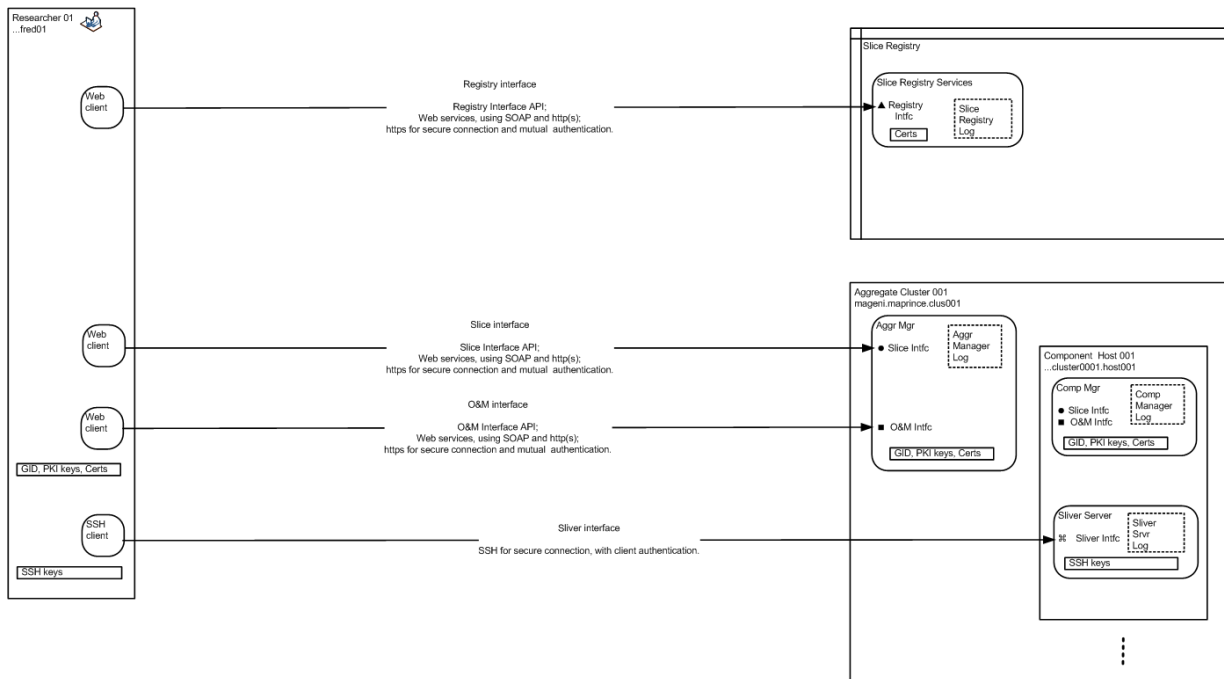
Sub-steps of instantiating a slice: A slice is first instantiated on a set of components with only best-effort resources. Later, given additional (perhaps guaranteed) resources for a certain amount of time

An experiment then "runs in" a slice.

## **8 Message Flow**

## 8.1 Message Flows in ProtoGENI

Researchers continually connect and communicate with Registries and Aggregates via defined interfaces and APIs. Aggregates supply resources that Researchers consume.



[ProtoGENI Message Flows]

## 8.2 Message Flows in PlanetLab

Principals periodically connect and communicate with Registries, Aggregates, Components, and Slice Managers through defined interfaces and APIs.

Researchers periodically connect and communicate with Registries, Aggregates, Components, and Slice Managers in order to acquire resources necessary to setup and execute experiments – Aggregates supply resources and Researchers consume resources.

Users interact with the Slice Manager which, in turn, contacts the registry to retrieve credentials and then invokes the Slice Interface on the aggregate to create and control the slice. The aggregate, rather than the users, interacts with the individual nodes. Current implementation uses a private interface to interact with the individual components.

Using the alternate slice manager, the Slice Manager contacts the registry to retrieve necessary credentials, contacts the aggregate manager to retrieve a ticket for each slice to

instantiate. The alternate slice manager then directly contacts the nodes to redeem tickets, and later to control the slices on the nodes. The alternate slice manager is responsible for ensuring that the slices persist across node failures.

## **9 Goals for Federation**

The GENI Control Framework Requirements highlight two goals for Federation on the Control Framework level.

The GENI control framework should provide for the inclusion of a wide variety of federated aggregates into a GENI suite whose native control framework is the GENI control framework, and another GENI suite whose native control framework is different than the GENI control framework.

The control framework should provide for the federation of a GENI suite with one or more suites that utilize the same control framework structure as the GENI suite. The control framework should provide for the federation of a GENI suite with one or more suites that do not utilize the same control framework structure as the GENI suite.

## **10 Definitions**

**Aggregate-** An aggregate is an object representing a group of components, where a given component can belong to zero, one, or more aggregates. Aggregates can be hierarchical, meaning that an aggregate can contain either components or other aggregates. Aggregates provide a way for users, developers, or administrators to view a collection of GENI nodes together with some software-defined behavior as a single identifiable unit. Generally aggregates export at least a component interface, i.e., they can be addressed as a component, although aggregates may export other interfaces, as well. Aggregates also may include (controllable) instrumentation and make measurements available. This document makes broad use of aggregates for

operations and management. Internally, these aggregates may use any O&M systems they find useful.

Clearinghouse - A clearinghouse is a, mostly operational, grouping of a) architectural elements including trust anchors for Management Authorities and Slice Authorities and b) services including user, slice and component registries, a portal for resource discovery, a portal for managing GENI-wide policies, and services needed for operations and management. They are grouped together because it is expected that the GENI project will need to provide this set of capabilities to bootstrap the facility and, in general, are not exclusive of other instances of similar functions. There will be multiple clearinghouses that will federate. The GENI project will operate the NSF-sponsored clearinghouse. One application of 'federation' is as the interface between clearinghouses.

Components - A component might correspond to an edge computer, a customizable router, or a programmable access point. A component encapsulates a collection of resources, including physical resources (e.g., CPU, memory, disk, bandwidth) logical resources (e.g., file descriptors, port numbers), and synthetic resources (e.g., packet forwarding fast paths). These resources can be contained in a single physical device or distributed across a set of devices, depending on the nature of the component.

Experiment - An experiment is a researcher-defined use of a slice; we say an experiment runs in a slice. Experiments are not slices. Many different experiments can run in a particular slice concurrently or over time.

Federation - Resource federation permits the interconnection of independently owned and autonomously administered facilities in a way that permits owners to declare

resource allocation and usage policies for substrate facilities under their control, operators to manage the network substrate, and researchers to create and populate slices, allocate resources to them, and run experiment-specific software in them.

**Resource -** Resources are abstractions of the sharable features of a component that are allocated by a component manager and described by an RSpec. Resources are divided into computation, communication, measurement, and storage. Resources can be contained in a single physical device or distributed across a set of devices, depending on the nature of the component.

**Slices -** From a researcher's perspective, a slice is a substrate-wide network of computing and communication resources capable of running an experiment or a wide-area network service. 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. A slice is the basis for resource revocation (i.e., shutdown). A slice is defined by a set of slivers spanning a set of network components, plus an associated set of users that are allowed to access those slivers for the purpose of running an experiment on the substrate. That is, a slice has a name, which is bound to a set of users associated with the slice and a (possibly empty) set of slivers.

**Slivers -** It must be possible to share component resources among multiple users. This can be done by a combination of virtualizing the component (where each user acquires a virtual copy of the component's resources), or by partitioning the component into distinct resource sets (where each user acquires a distinct partition of the component's resources). In both cases, we say the user is

granted a sliver of the component. Each component must include hardware or software mechanisms that isolate slivers from each other, making it appropriate to view a sliver as a 'resource container.'