G E N I

Global Environment for Network Innovations

ORCA GENI Control Framework Overview

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

January 14, 2009

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

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 ORCA GENI control framework being implemented for Spiral 1, for use in Cluster D. 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 ORCA team is underway, but has not yet been completed.) It provides a description of the ORCA 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 the "ORCA Technical Note: Guests and Guest Controllers", available at http://www.cs.duke.edu/nicl/pub/papers/control.pdf .

Some of the material in this document is taken from summaries and notes provided by Jeff Chase.

1.2 Context for this Document

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

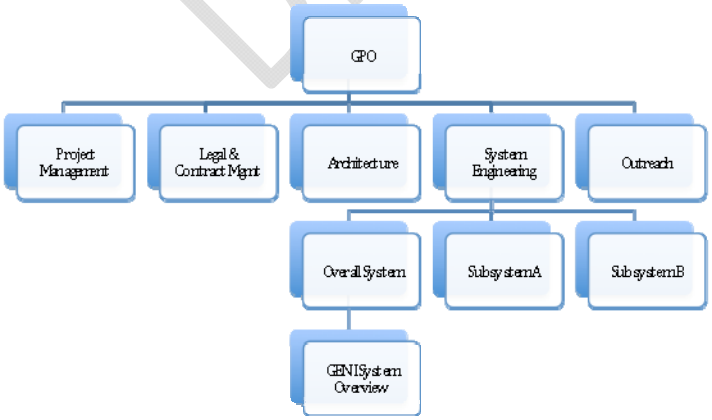

Figure 1-1. This Document within the GENI Document Tree.

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
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	SHARP: An Architecture for Secure Resource Peering, 2003, by Yun Fu, Jeffrey Chase, et.al. http://www.cs.ucsd.edu/~vahdat/papers/sharp-sosp03.pdf
N/A	Sharing Networked Resources with Brokered Leases, 2006, by David Irwin, Jeffrey Chase, et.al. http://portal.acm.org/citation.cfm?id=1267377
N/A	ORCA Technical Note: Guests and Guest Controllers, 2008, by Jeff Chase http://www.cs.duke.edu/nicl/pub/papers/control.pdf
N/A	ORCA references: http://nicl.cod.cs.duke.edu/orca/
N/A	Towards an Autonomic Computing Testbed, 2007, by Aydan Yumerefendi, et. al. http://www.cs.duke.edu/~shivam/hotac07.pdf
N/A	Understanding WS_Federation http://xml.coverpages.org/UnderstandingWS-Federation20070528.pdf
N/A	Weighted Fair Sharing for Dynamic Virtual Clusters, 2008, by Laura Grit and Jeff Chase, et.al. http://portal.acm.org/citation.cfm?id=1375521
N/A	Secure Control of Portable Images in a Virtual Computing Utility, 2008, by Ionut Constandache, Aydan Yumerefendi and Jeff Chase http://www.cs.duke.edu/~ionut/2008_vmsec.pdf

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 ORCA material provided by Jeff Chase, and following structure from Control Framework Requirements document.
01.2	1/14/09	H. Mussman	Updated with small changes.

01.3			
01.4			

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.

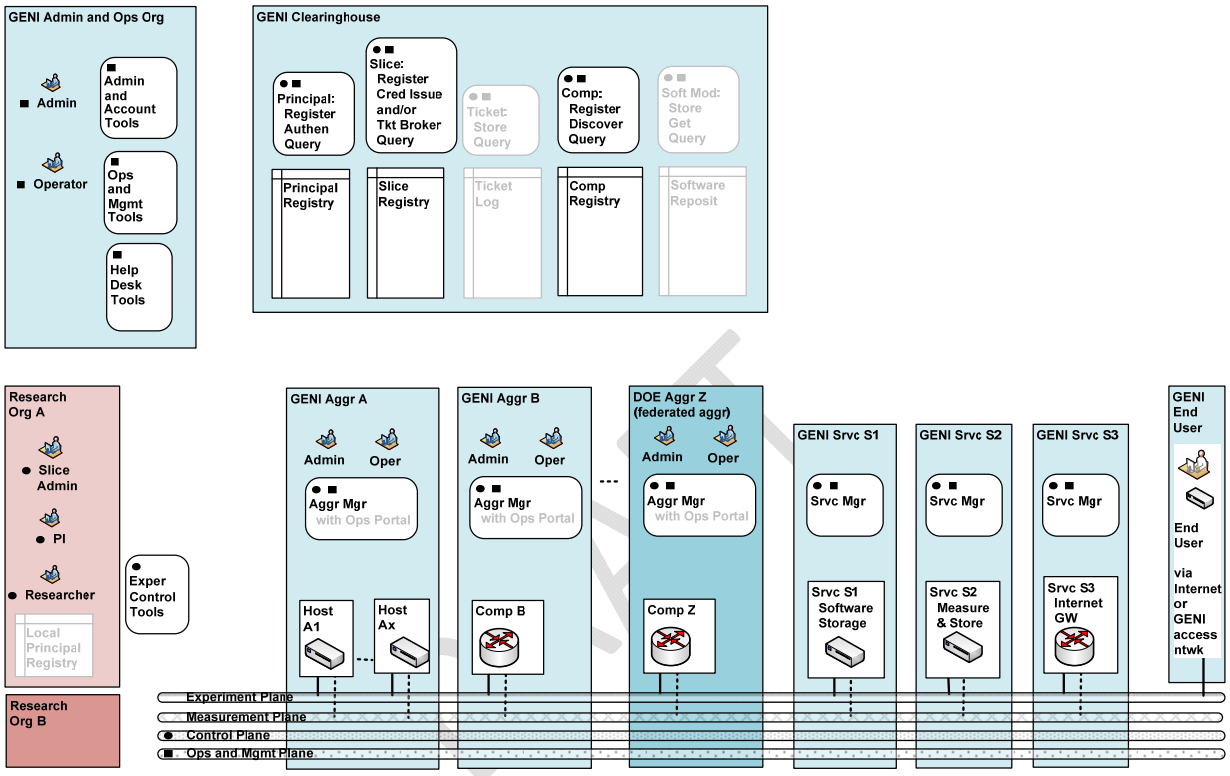

Figure 2-1. GENI System Diagram.

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).

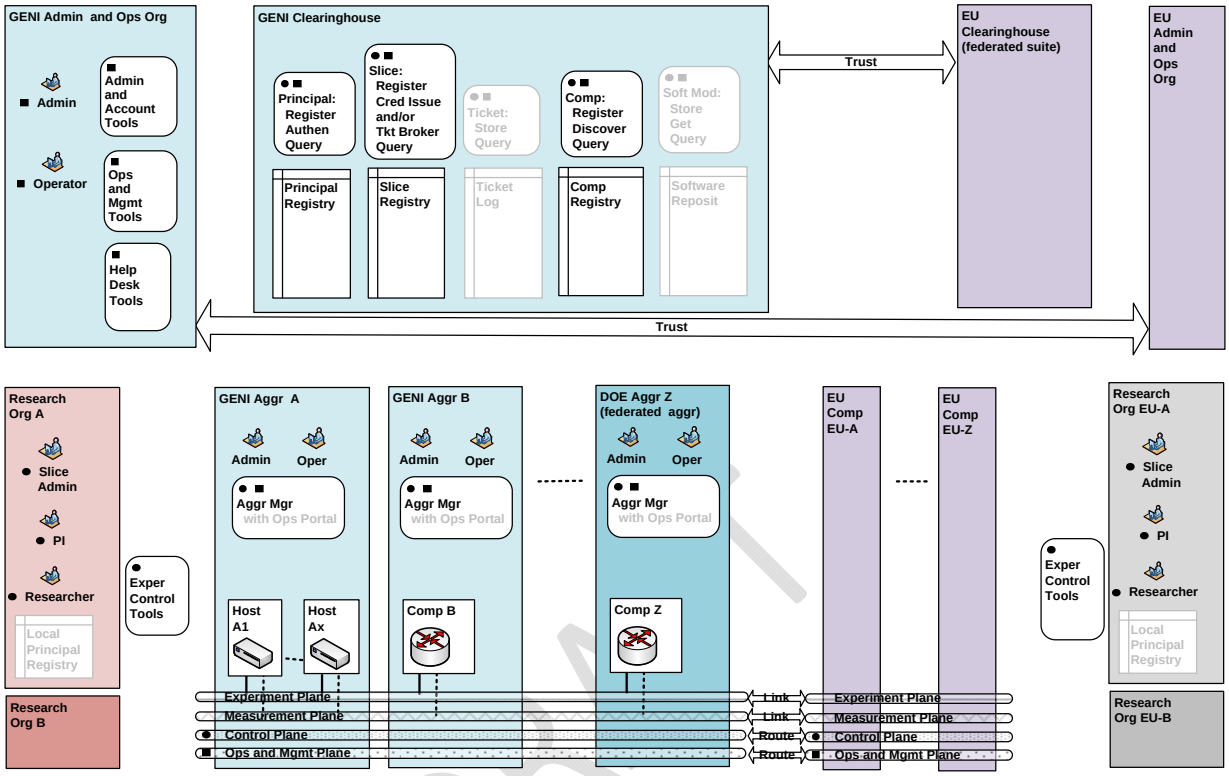

Figure 2-2. System Diagram with Federated Infrastructure Suites.

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.

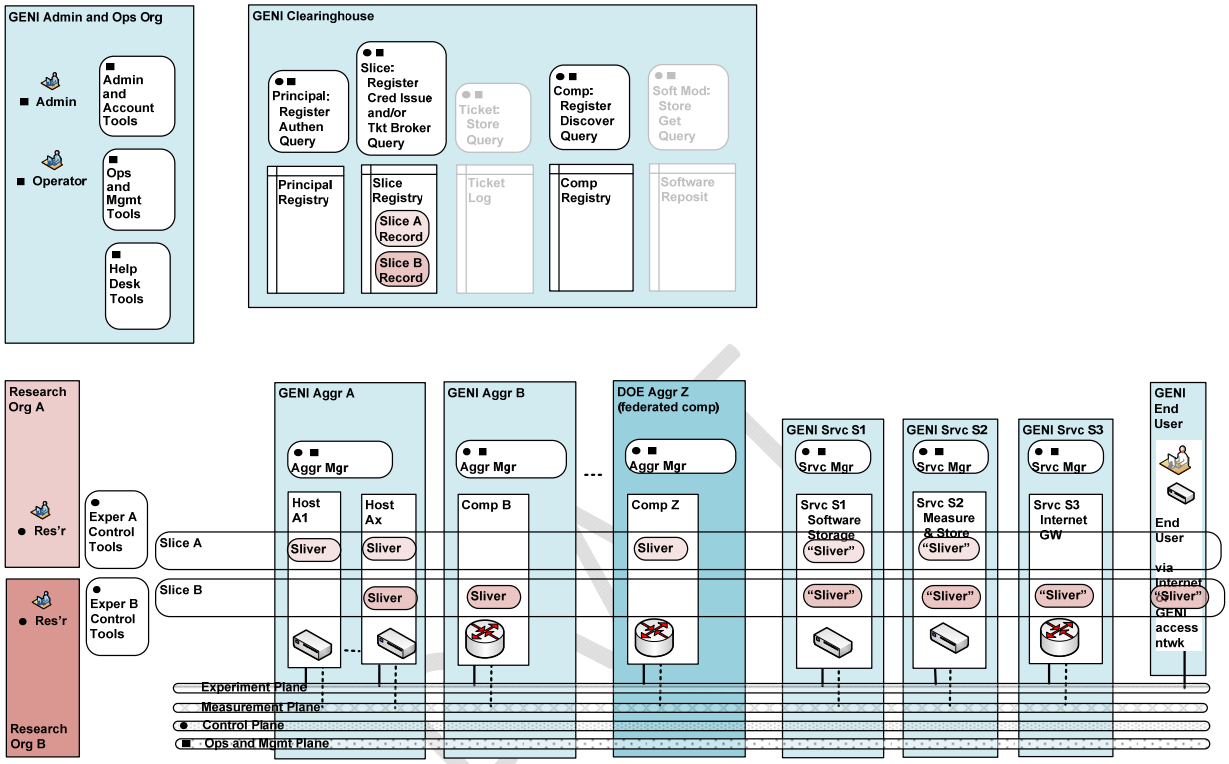

Figure 2-3. Two GENI Slices.

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:

- ORCA, 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/ORCA .
- 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.

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 ORCA GENI-based control framework implementation.

4 ORCA GENI Control Framework Structure

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

- A Broker, which provides most Clearinghouse functions.
- One or more Site/Domain Authorities (Aggregate Managers), which control Components, e.g., an array of hosts at a site, or a network domain.
- One or more Service Managers (Experiment Control Tools) which are used by the Researchers to setup and manage slices.
- Principals, such as Administrators, PIs, Operators, and Researchers, that connect with the Broker, Site/Domain Authorities and Service Managers.
- One or more Identity Providers, which vouch for Principals.

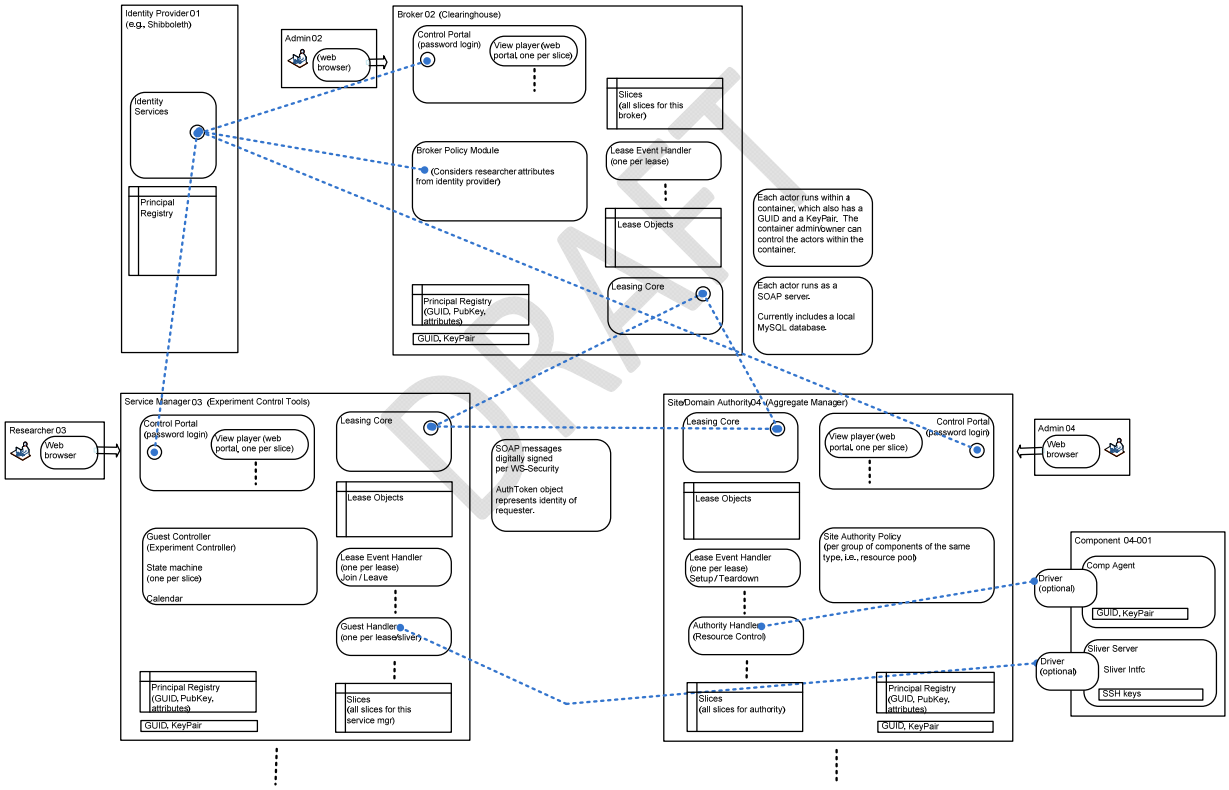

Figure 4-1. ORCA GENI Control Framework Structure.

The Open Resource Control Architecture (ORCA) emphasizes emphasizes modular policies for resource management and slice adaptation. Orca derives from the SHARP architecture for resource leasing contracts with accountable ticket brokering. See SHARP: An Architecture for Secure Resource Peering, available at http://www.cs.ucsd.edu/~vahdat/papers/sharp-sosp03.pdf .

A summary of ORCA can be found in a technical note available at http://www.cs.duke.edu/nicl/pub/papers/control.pdf and from the website http://nicl.cod.cs.duke.edu/orca/ .

Some of the current ORCA implementations are described in Towards an Autonomic Computing Testbed available at http://www.cs.duke.edu/~shivam/hotac07.pdf .

Note that the ORCA GENI control framework is not based upon earlier GENI efforts (including "GMC Specifications," "GENI Distributed Services," and "GENI Engineering Guidelines.") or the "Slice-based Facility Architecture" (SFA) at http://svn.planet-lab.org/attachment/wiki/GeniWrapper/sfa.pdf .

4.1 Actors

The core of the Spiral 1 implementation is a Java-based resource leasing toolkit called Shirako. Shirako is the basis for Java reference implementations of a GENI clearinghouse, aggregate managers, and experiment control tools.

These GENI control framework entities match the respective Orca/Shirako actor roles: broker, domain authority, and service manager (guest controller).

The actors (automated control servers) run as SOAP servers exchanging digitally signed messages (WS-Security); see Understanding WS_Federation at http://xml.coverpages.org/UnderstandingWS-Federation20070528.pdf.

Each actor also has a web control portal interface for the user or operator.

Every actor maintains a similar basic set of data structures. The tables are maintained by the leasing core, i.e., the code that operates on them is in the core.

- a list of slices known to it ("slice registry"?)

- a list of lease objects (aka "reservations") known to it. We call them "lease" or "reservation" objects even if the contract has not yet been consummated yet by issuing a lease contract, i.e., a "ticket" corresponds to a lease object in the "ticketed" state.

- a list of identities (AuthToken) of other entities that are known to it ("principal registry"?)

The Shirako leasing core is basically a common implementation of this data structure:

- Brokers track the slices that request resources from them, and the tickets issued to those slices.

- Authorities track the slices that have leased resources from them, and the leases issued to those slices.

- Service Managers only track the slices and leases that they create and initiate.

Of course there are lots of places where the code has variants appropriate to the different actors...it is not all common. The differences are mostly in the classes that implement the lease state machines, which have different states and transitions for each actor.

And also in the plugin modules.

But there are many symmetries. For example, "handler" plugins are invoked from the lease object state machine on particular lease state transitions. The same code in the leasing core upcalls authority event handlers (setup/teardown) as calls SM event handlers (join/leave).

These handlers are invoked when resources enter and leave a slice.

None of the tables is really a "registry" in the sense that it contains all the slices or leases in the system. Actors only track what they know about. If we choose to use a centralized broker (clearinghouse), then the slice table in that broker would serve as a "slice registry" in that it would be the only table listing all current slices. But if there were multiple brokers, there would not necessarily be a complete "slice registry" anywhere

4.2 Brokers

The ticket broker service provides all essential clearinghouse functions, and issues all tickets to experiments. All site/domain authorites delegate splittable tickets to the broker service, and attempt to honor any tickets issued by the broker.

The ticket broker service maintains a principal registry containing the public keys of identity providers and users registered by the clearinghouse operator. The service accepts any user with a registered public key, or bearing an X.509 certificate endorsed by a registered identity provider.

The ticket broker policies arbitrate resource shares among user identities grouped by attributes. Resources include various sizes of virtual machines and connectivity options, e.g., between BEN sites.

The strong broker role in the clearinghouse makes it possible to experiment with policies for coordinated resource allocation.

The ticket broker service is also the primary means to discover resources in the prototype, i.e., a slice controller discovers a component by requesting tickets for resources by attributes, and receiving ticket for component aggregates matching those attributes.

Broker (clearinghouse) federation is not supported. Orca enables loose federations of substrate providers that delegate splittable tickets to a given clearinghouse for subsets of their resources over specific intervals of time, as directed by their operators. A given substrate provider may offer resources to multiple clearinghouses.

4.3 Site/Domain Authorities and Components

All components are aggregated and controlled by a site/domain authority. The site/domain authority subsumes the role of Management Authority for the substrate resources that it controls. For Spiral 1 we plan to demonstrate site/domain authorities for edge clusters, an optical connectivity provider (RENCI's Breakable Experimental Network, BEN), sensor testbeds, and a mobile testbed (DieselNet).

The edge clusters use virtual machine slivering (e.g., based on Xen hypervisor), in conjunction with a local virtual cluster manager (e.g., Eucalyptus). Node slivers (virtual machines) can boot any named image accepted by the local cluster manager.

As an aside, to simplify code, other resource contract representations are shoehorned into the same framework, e.g.:

- An authority holds a "slice" containing its resource inventory (in geni-speak, the resources assigned to an aggregate manager by a management authority).

- Authorities issue tickets to brokers to delegate control over resources. A delegation is represented on both sides as a "lease object" within some slice representing the inventory holdings and/or inventory outlays.

4.4 Component Agents

Each ORCA GENI component includes a native Component Agent to setup, configure and control the component. In some cases, a component will assign one or more "sliver servers" to a lease.

Controllers and handlers often invoke external programs or scripts that run configuration actions directly on a controlled component (e.g., a guest node). To this end we have developed a framework for dynamically installable driver packages that run under a component agent (node agent) on the controlled resources, and are invoked from a controller or handler.

Handlers invoke guest driver interfaces to attach and detach allocated resources to and from the guest. A guest driver is a wrapper or package of scripts that defines a set of control actions on a guest instance. A guest driver exports standard actions to instantiate and configure the guest, probe its status, and also to attach and detach managed resources (servers, storage) to and from the service, and to notify it of VM resizing or migration.

Many resource units can be viewed as nodes. Examples include physical servers, virtual machines, or any resource that can be controlled by scripts or programs running at some IP address, such as a domain-0 control interface for a virtual machine hypervisor. Node drivers run under a standard node agent that runs on a node, e.g., in a guest VM, a control VM (Xen dom0), or a control node for some other substrate component. A node driver is a packaged set of actions that run under the node agent to perform resource-specific configuration on a node.

The driver approach enables continuous dynamic control during operation, and it generalizes to guests that are installable after booting as packages. It also enables dynamic installs and upgrades of guest drivers, and asynchronous invocation of arbitrary driver programs or scripts.

The node agent accepts signed requests from an authorized actor on the network, e.g., using SOAP/WS-Security or XMLRPC, including requests to install, upgrade, and invoke driver packages. The guest join handler may load the guest driver and guest code itself onto the allocated nodes from an external source, such as an external package repository or Web server, or a shared service.

The result of each driver method invocation is also a property list, which flows back to the handler that invoked it. The property list conveys information about the outcome of a driver invocation (e.g., exit code and error message). The handler determines how to represent any result properties in the lease state. An error may trigger a failure policy on the service manager that resets a resource or re-assigns the resource to an equivalent machine.

4.5 Service Managers

The ORCA GENI control framework includes Service Managers that are used by Researchers to setup and manage slices.

A Service Manager includes a Guest Controller and a Calendar in addition to the Leasing Core which provides interfaces to the other actors. The query method sends a query request to another actor and the demand method prepares a new lease object to generate a ticket request to a broker.

The following are the general steps a guest controller goes through to formulate and demand a request.

1. Track guest status and translate into resource demand. The guest controller learns the state

of the guest to make decisions about how to service its resource needs. This is application-specific and the guest controller may implement it in a variety of ways.

2. Track resource status. The guest controller inspects the current state of its resource holdings and their expiration times, which are indexed by calendar classes in the toolkit. Since leases are dependent on time, the calendars provide a way to store and query present and future resource holdings. They also provide convenient methods to query lease objects by their states.

3. Formulate resource requests. The guest controller formulates its requests as ResourceLease objects and sets its desired term, number of resource units, and resource types for each one (Section ??). To define additional attributes on the request (e.g., request exibility), the controller attaches request properties to the ResourceLease. These express additional constraints, degrees of freedom, or objectives (such as resource value) to a broker policy to guide resource selection and allocation.

4. Issue the request. Once the guest controller formulates its requests, it submits it using the demand method. The service manager shell verifies that each request is well-formed and queues it for transmission to the selected broker.

4.6 Principals

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

- Principal administrators, who act for the ORCA GENI suite or a research organization, and are responsible for principal records and the authentication of principals.
- Authority administrators, who act for the ORCA GENI suite or an owning organization, and are responsible for authority policies, etc.
- Broker administrators, who act for the ORCA GENI suite or a research organization, and are responsible for broker policies, etc.
- 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 ORCA GENI suite or an owning organization, and are responsible for operations and management functions within an aggregate (or component).
- Researchers, who utilize the ORCA GENI suite for running experiments, deploying experimental services, etc.

4.7 Identity Providers

The GENI ORCA control framework includes one or more Identity Providers which vouch for Principals, and can provide attributes for certain principals, e.g., for researchers.

4.8 Actor Authentication and Authorization

In the ORCA GENI control framework, authentication and authorization are based on digitally signed messages (WS-Security) and the Java Cryptography Architecture (e.g., keystore files). See Figure 4-2.

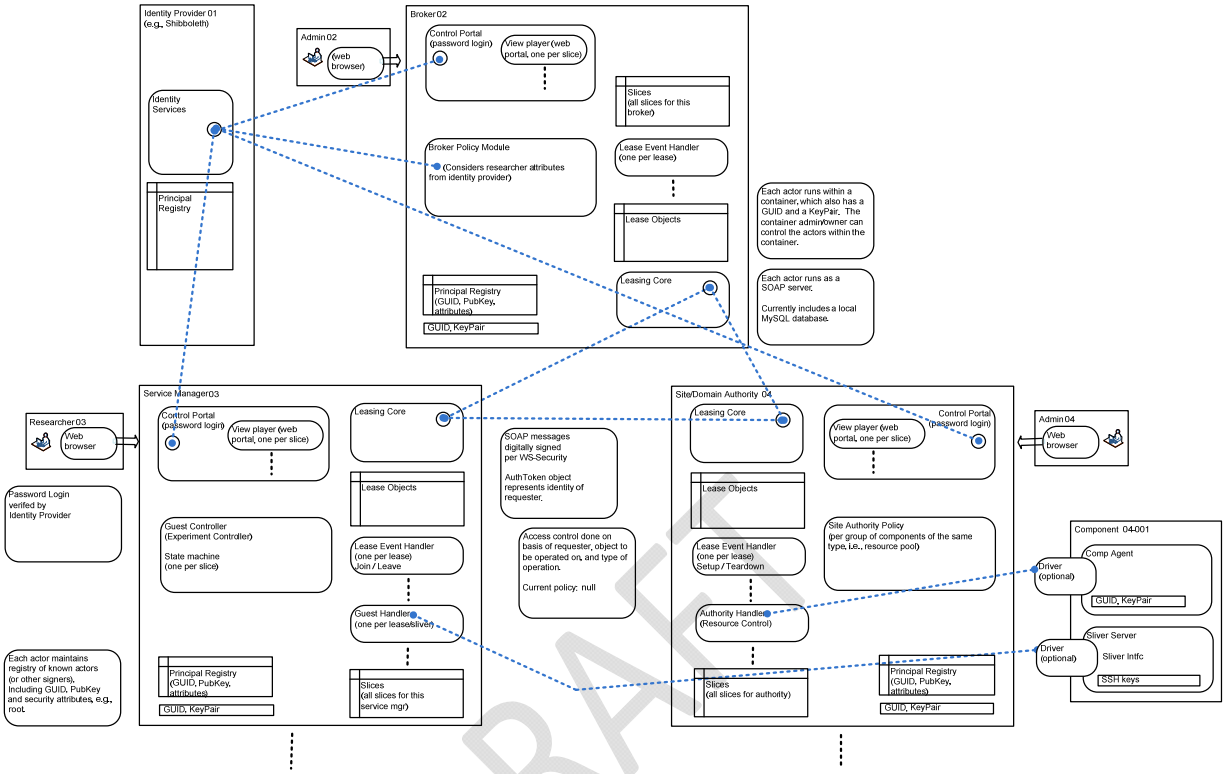

Figure 4-2. ORCA GENI Actor Authentication and Authorization.

Every actor is named by a GUID. Every actor possesses a keypair for authentication. Each actor has access to a registry of the GUIDs and public keys of other actors that are known to it. Actors sign their messages with their private keys, and authenticate messages based on their knowledge of the sender's public key. We use the term "principal" to refer to an identity bound to a public key. [Note that a principal could wield multiple public keys.]

Each actor runs within a container, which also has a keypair and GUID. A container administrator (owner) can control the actors within the container.

An actor's keypair and key registry comprise its "security configuration". The security configuration files reside in the "runtime directory" of the actor's container. The file names are derived from the actor's GUID, so the files persist and can be retrieved by GUID. Most aspects of security configuration are fully automated, but there are also manual tools available for a container owner to generate, view, and edit security configurations.

Each actor assigns some level of trust to other actors/principals on the basis of public key. Actors learn of public keys and assign trust to them in at least two ways. First, a container owner can use the configuration tools to register principals (public keys) for various levels of trust and access with the actors in a container. For example, a broker operator might register public keys of guests (service managers to be precise) that are authorized to request resources from that broker. Second, actors may

assign trust to a principal based on an endorsement of its public key from some other actor. For example, a ticket issued by a broker endorses the public key of a guest to an authority.

The actor's GUID and keypair are typically generated automatically when the actor is created. The code that instantiates an actor and binds it to a GUID and keypair is part of the "boot" code. The boot code runs to populate a freshly instantiated container with actors specified in a container configuration file. The same code also runs when an actor is created dynamically, e.g., through the Web portal.

For more control, a container owner may use the tools to generate the GUID and security configuration by hand in advance. In this case the configuration file or portal request may specify a GUID. [Note/Q: is it possible to hardwire an actor's public key, and install its private key in its security configuration.]

When an actor receives a request, the message receive proxy obtains a reference to an AuthToken object that represents the identity of the requester. The requester is authenticated by means of the public key associated with the signed request. The AuthToken of the requester is available to the request-handling code for the access control policy.

The request handling code upcalls an "auth" module that implements the access control policy for the requesting principal (subject), the object to be operated on and the type of operation. The current implementation uses the "null" access control policy (just say yes).

4.9 Message Flows

There are three types of message flows defined in the ORCA GENI control framework:

1) Between Leasing Cores in the three types of actors. The state for a brokered lease spans three interacting state machines, one in each of the three actors involved in the lease: the service manager that requests the resources, the broker that provisions them, and the authority that owns and assigns them. Each lease object has local state variables that represent the actor's view of the lease state. The lease state machines govern all functions of the core. State transitions in each actor are initiated by arriving requests or lease/ticket updates, and by events such as the passage of time or changes in resource status.

2) Between Guest and Authority Handlers within actors and Drivers that control Components. Handlers invoke guest driver interfaces to attach and detach allocated resources to and from the guest. A guest driver is a wrapper or package of scripts that defines a set of control actions on a guest instance. A guest driver exports standard actions to instantiate and configure the guest, probe its status, and also to attach and detach managed resources (servers, storage) to and from the service, and to notify it of VM resizing or migration.

3) Between Principals and the actors, via a web portal defined within the actor. These enable users to monitor and interact with guests and controllers during an experiment. For example, the guest controller plugins can implement a graphical interface to set attributes or parameters for the guest, e.g., to modulate a workload generator. Views plug-ins typically are server-side Web templates (e.g., in Velocity) that run at the Web portal, which could run in a different JVM from the actor. They may also include applets or other client-side code.

4.10 Slices

From a Researcher's perspective, a slice is an interconnected set of reserved resources, or slivers, on heterogeneous substrate aggregates (components).

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.

At each broker, on initial request of a new slice, the broker assigns a GUID to the slice, and attaches a slice property whose value is an X.509 cert containing the requesting guest's public key and the slice GUID. The guest controller should use this GUID as the GUID of the slice, rather than a GUID that it creates itself. The guest controller should pass the slice certification property through on all ticket and redeem requests for this slice. Brokers and authorities should check the certification when they first learn of a slice, and save it with the slice. [Note: this is not a required function, it simply provides compliance with the GENI CFA notion of a "slice authority".]

Each slice is also given a Global Name (GNAME). By whom? Is it part of the GUID?

4.11 Leases and Tickets

In ORCA GENI, actors retain and cache lease state in memory. Each actor has a local lease object to represent each lease, regardless of its state. There are different lease classes with interfaces tailored to the different actor roles. We summarize the lease interface using a fictional class called ResourceLease.

There are a few state elements that are common to all leases. These are defined for direct use by the core rather than incorporated into the properties:

1) State. The state for a brokered lease spans three interacting state machines, one in each of the three actors involved in the lease: the service manager that requests the resources, the broker that provisions them, and the provider that owns and assigns them. Each lease object has local state variables that represent the actor's view of the lease state. The lease state machines govern all functions of the core. State transitions in each actor are initiated by arriving requests or lease/ticket updates, and by events such as the passage of time or changes in resource status.

2) Term. The interval of time over which the lease is valid. In practice there are multiple term variables used during negotiation. When a guest controller formulates a resource request, the term indicates the preferred start time and duration of the lease. The start time may be in the future for an advance reservation; the ResourceLease's term can also be modified to alter the start time, and duration (e.g., on lease extension) before the next request or before a ticket is granted.

3) Type. Resources are typed, and each lease pertains to resources of a single type. An actor may query the space of available resource types on a broker. The resource type does not change during the lifetime of a lease. However, various resource properties associated with the type may be mutable. These are often called \subtype" properties, and they represent virtual sliver attributes that may take different values on different instances of the type (e.g., sizing attributes for a virtual machine such as the share of CPU power available to it).

4) Unit count. Each lease pertains to a block of units with identical resource attributes, e.g., a cluster of virtual machines with identical size.

4.12 Property Lists

Property Lists are used in ORCA GENI to specify resources, including the functions provided by GENI RSpecs. Property Lists are [key, value] pairs that are attached to leases. There are four types:

1) Request properties (from srvc mgr to broker)
2) Resource type properties (common to type, from broker to srvc mgr)
3) Configuration properties (for setup, from srvc mgr to authority)
4) Unit properties (as assigned, from authority to srvc mgr)

Resource properties specify attributes that are common to all resources of a given type. Brokers attach resource properties to each ticket. The properties are readable to all downstream plugins on the guest and authority.

Configuration properties direct how the authority instantiates or configures resources on setup, or modified on an extend. The guest controller sets these properties on the lease object before the lease is ticketed or before it is redeemed or extended. The core passes the configuration properties through to the authority at redeem or extend time, where they are readable to the authority-side plugins.

Unit properties define additional attributes for each resource unit assigned. Authority plugins attach unit properties to each resource unit. The core transmits them to the service manager with each lease and then through to the guest plugins.

To reduce the overhead to transmit, store, and manipulate lease properties, the messaging stubs (proxies) transmit only the property set associated with each protocol operation, and receivers ignore any superfluous property sets. Most unit properties and type properties define immutable properties of the resource that service managers and authorities may cache and reuse freely. Finally, the slice itself also includes a property set that all leases within the slice inherit.

4.13 Authorization and Policies

GENI ORCA emphasizes modular policies for resource management and slice adaptation.

The Broker includes a Broker Policy Module that is used when it assigns Tickets to Researchers. In this assignment, it can consider a researcher's attributes, as received from an identity provider.

A Site/Domain Authority includes a Authority Policy Module that defines the policy for assigning resources in a particular resource pool, a group of components of the same type.

This Authority Policy Module can include a scheduler to control the quantites of resources shared over time in a fair manner across multiple, competing customers; see Weighted Fair Sharing for Dynamic Virtual Clusters at http://portal.acm.org/citation.cfm?id=1375521 .

4.14 Slivers

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 using the Service Manager and a Guest Handler 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. ORCA 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 Guest Handler.

A more general approach to downloading an image into a server is described in "Secure Control of Portable Images in a Virtual Computing Utility" at http://www.cs.duke.edu/~ionut/2008_vmsec.pdf .

5 Principals in the ORCA GENI Control Framework

5.1 Identification

Each Principal (user) in ORCA GENI, such as a Researcher, has a unique identifier, which is used when it connects to an actor, such as a Service Manager.

(The Principals are different from the Actors; see Section 4.8 for Actor Authentication and Authorization)

Question: What is this identifier? Can it be different for different groups of Principals (Researchers)?

5.2 Registration

A Principal (user) in ORCA GENI, such as a Researcher, may be registered in an actor (e.g., a Service Manager) by the container owner. (Could this be a Research Organization?).

A Principal (user) in ORCA GENI, such as a Researcher, may be registered in their local organization, and known to an Identity Provider.

Question: How does this work?

5.3 Authentication

As a ORCA GENI principal (e.g., Researcher) connects to an actor, it is authenticated by the local Principal Registry or an associated Identity Provider.

Question: How does this work?

6 Site/Domain Authorities and Components in ORCA GENI Control Framework

6.1 Identification

Each Authority and Component in ORCA GENI has a unique Global Identifier (GUID) that does not change, and a Public Key that is bound to it by a certificate; see Section 4.8.

Question: Who creates the GUID, and what does it include?

6.2 Registration

Each Authority is "registered" with the Broker when it delegates splittable Tickets to the Broker.

Question: How is it done? Where are these tickets held in the Broker?

Question: How are Components "registered" with an Authority?

6.3 Resource Allocation

An Authority implicitly delegates resources to the ORC GENI suite when it delegates splittable Tickets to the Broker.

Question: Can these Tickets indicate times as well as resources? Do they need to be replaced after the Broker issues Tickets?

7 Slices in ORCA GENI Control Framework

7.1 Identification

At each broker, on initial request of a new slice from a component manager, the broker assigns a GUID to the slice, and attaches a slice property whose value is an X.509 cert containing the requesting guest's public key and the slice GUID. See Section 4.10.

Each slice is also given a Global Name (GNAME).

Question: What is the GUID?

Question: Global Name (GNAME)? Who creates it?

7.2 Registration

The broker, on initial request of a new slice, creates the slice.

Then:

- Brokers track the slices that request resources from them, and the tickets issued to those slices.

- Authorities track the slices that have leased resources from them, and the leases issued to those slices.

- Service Managers only track the slices and leases that they create and initiate.

Question: Is there a way to predefine a slice before an initial request for a new slice is received from a component manager?

Question: Is there a way to define the principals that can use a slice? That can administer a slice?

7.3 Ticket Broker

The ticket broker service issues all tickets to experiments, and associates each ticket with a slice.

8 Experiment Setup in ORCA GENI Control Framework

The ORCA GENI control framework provides all of the basic functions necessary for a GENI researcher to setup an experiment. See Figure 8-1.

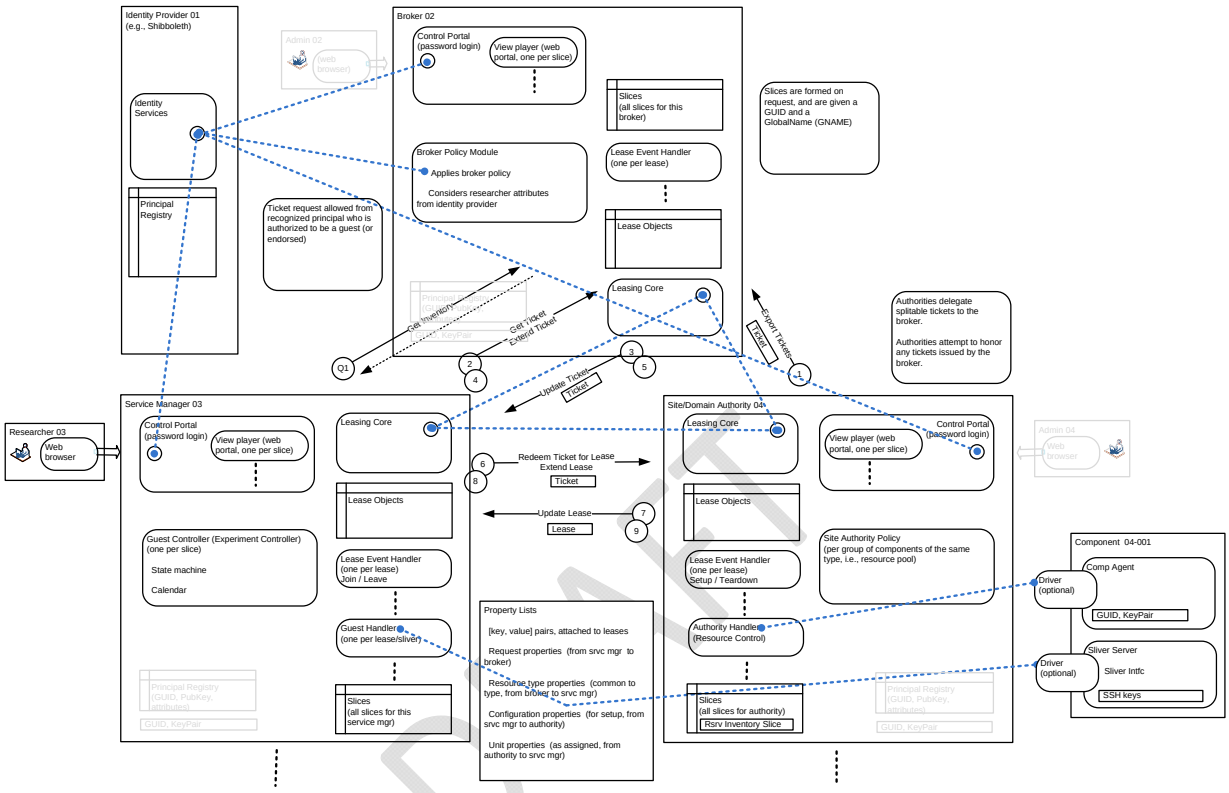

Figure 8-1. ORCA GENI Experiment Setup.

One more thing that is relevant to fig #2. "UpdateTicket" and "UpdateLease" are SOAP calls to pass tickets and leases to a service manager. The responses to the (get) "ticket" and "redeem" (lease) calls from the service manager just indicate whether the server accepted the

request (i.e., agreed to consider it). The actual ticket or lease

comes in a separate Update* call later. This asynchrony is important because it allows the response to be arbitrarily delayed. The requester (service manager) can decide whether it wants to time out and cancel, but at least it knows the request is under consideration. Many settings may impose a delay before the request is satisfied (or denied), e.g., for delay-tolerant networking, intermittent connectivity to the resources (dieselnet), resource-optimizing policies, etc.

Update* also serves as a notification interface. Update* calls may also be initiated unilaterally by the resource server (broker or authority).

Our brokers actually never use this, but the authority may use it, e.g., to notify that it has detected that some component has failed, or it was forced to take it out of service or otherwise unilaterally revoke the lease (or some part of it) and reclaim the component.

These protocols may be similar to WSRF, which has explored some of the same ground.

8.1 Resource and Topology Discovery

The ORCA GENI control framework allows a Researcher using a Service Manager to discover all of the resources available to them by sending a query to the Broker. See Step Q1 in Figure 8-1.

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

Question: How can they discover their interconnection topology?

8.2 Resource Sharing

In ORCA GENI, each Site Authority and Component provides for resource sharing among multiple researchers, referencing multiple slices, and assigns each researcher their own sliver or slivers.

8.3 Resource Authorization and Policy Implementation

The ORCA GENI control framework allows a Broker to authorize the assignment of resources to a Researcher using a Service Manager and referencing a particular Slice, following Broker policies.

See steps xx through yy in Figure 8-1.

8.4 Resource Assignment

The ORCA GENI control framework allows an Authority to assign resources to a Researcher using a Service Manager and presenting a valid Ticket, and apply Authority policies as this is done.

See steps xx through yy in Figure 8-1.

8.5 Component Programming

The ORCA GENI control framework allows a Researcher using the Service Manager and a Guest Handler to login to an assigned sliver (Component), load code, and then boot it, etc.

See Section 4.14.

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 ORCA 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 ORCA 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.

The ORCA GENI control framework allows a Researcher using the Service Manager to learn about connections and do this. See section x.

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 ORCA GENI.)

9 Experiment Execution in ORCA GENI Control Framework

9.1 Experiment Control

When a Researcher, associated with a designated slice, has been assigned resources in components for an experiment, the ORCA 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 9-2.

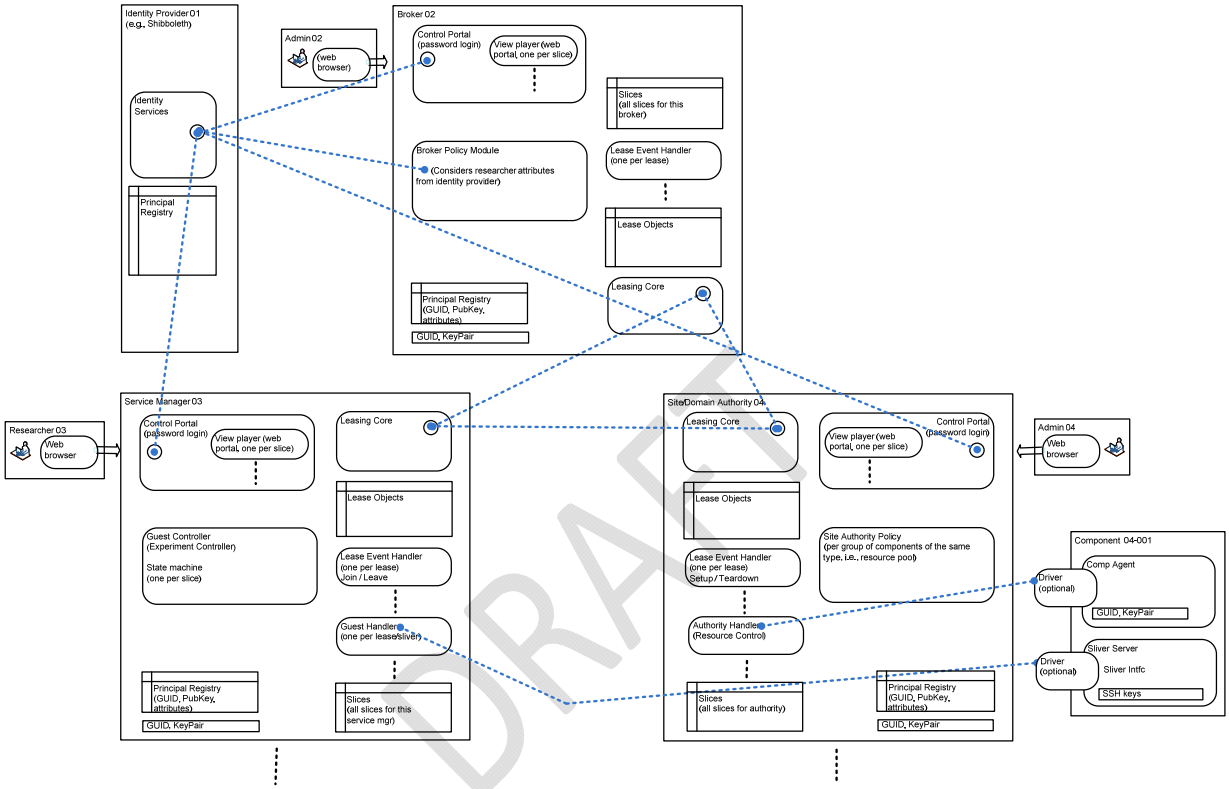

Figure 9-1. ORCA GENI Experiment Control.

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 ORCA 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.

The ORCA GENI control framework provides many ways to do this. See section xx.

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 ORCA GENI. Would there be an easy way to do this?)

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 ORCA GENI. Would there be an easy way to do this?)

10 Federation in ORCA GENI Control Framework

10.1 Federated Site/Domain Authorities and Components

A ORCA 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.

How?

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.

(Not yet defined in GENI ORCA.)

How?

11 ORCA GENI Cluster D Spiral 1 Implementation

11.1 Start of Spiral 1

At the beginning of Spiral 1, the ORCA GENI Cluster D implementation includes one Broker; one (or more?) Service Managers and multiple Site/Domain Authorities. See Figure 11-1.

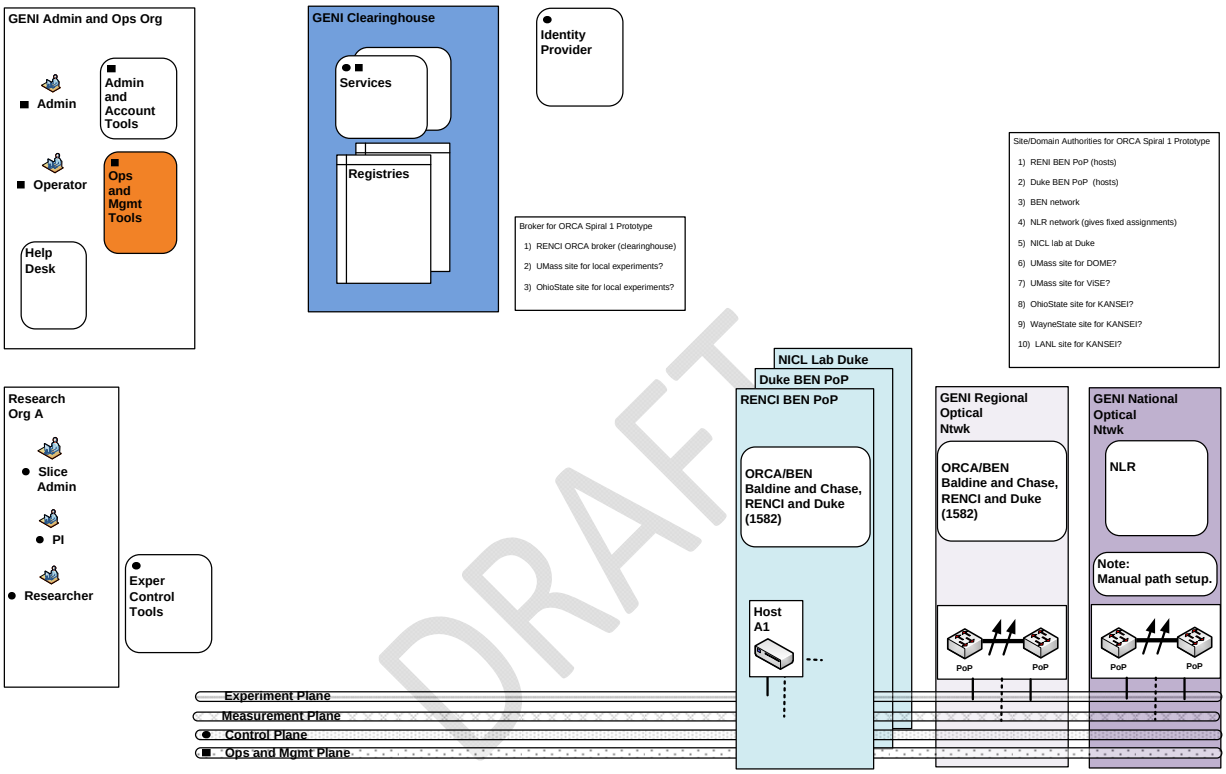

Figure 11-1. Start of ORCA GENI Cluster D Spiral 1 Implementation.

Question: How many Service Managers are included?

11.2 Completion of Spiral 1

At the completion of Spiral 1, the ORCA GENI Cluster D implementation includes one Broker; multiple Service Managers and multiple Site/Domain Authorities. See Figure 11-2.

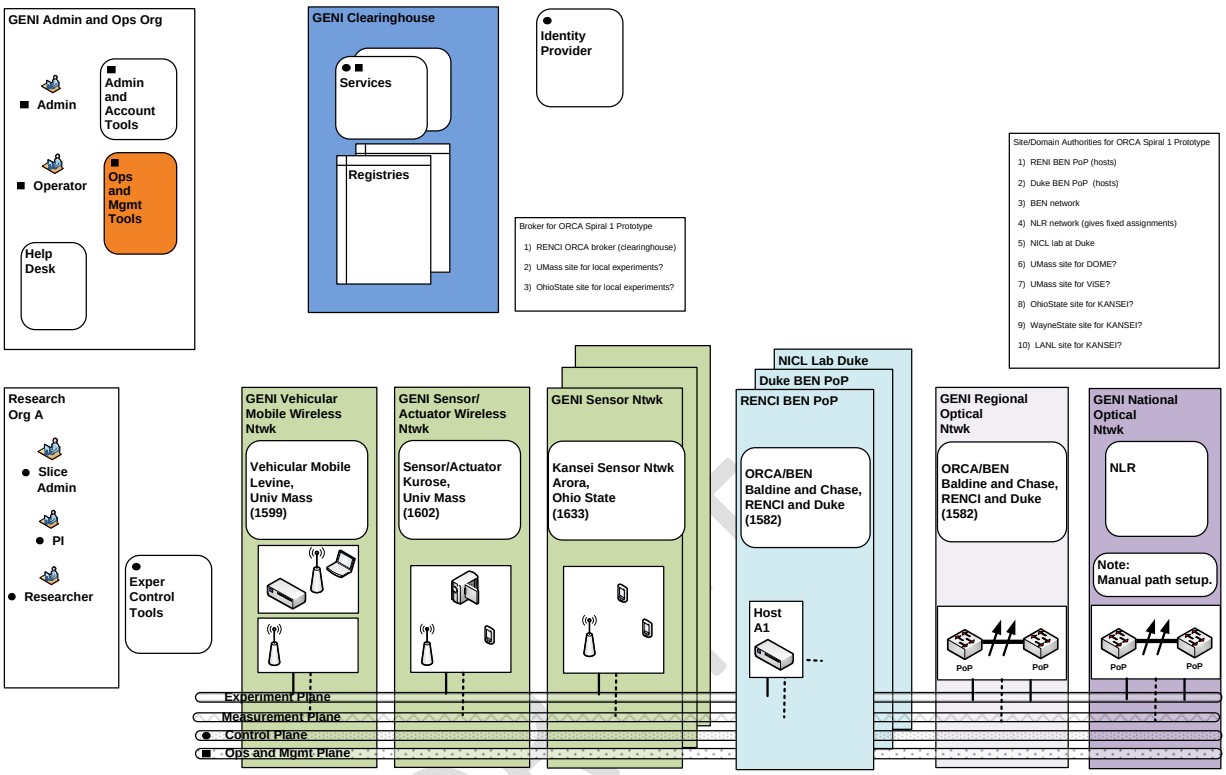

Figure 11-2. ORCA GENI Cluster D Implementation.