CAGRID 1.1 USER'S GUIDE

Center for Bioinformatics

September 14, 2007

Credits and Resources

caGrid Development and Management Teams		
Development	**Support (Systems, QA, Documentation)**	**Management**
Scott Oster (Lead Architect)[1]	Aynur Abdurazik[9]	Avinash Shanbhag (Product Manager)[9]
Stephen Langella[1]	Ye Wu[9]	Michael Keller[8]
Shannon Hastings[1]	Todd Cox[9]	Arumani Manisundaram[8]
David Ervin[1]	Wendy Erickson-Hirons[7]	
Tahsin Kurc[1]	Chet Bochan[11]	
Joel Saltz[1]	Vanessa Caldwell[11]	
Ravi Madduri[2]	Craig Fee[11]	
Ian Foster[2]	Alan Klink[11]	
Patrick McConnell[3]	Gavin Brennan[11]	
Joshua Phillips[5]		
Vijay Parmar[10]		
[1]Ohio State University - Biomedical Informatics Department	[2]University of Chicago/Argonne National Laboratory	[3]Duke Comprehensive Cancer Center
[4]ScenPro, Inc.	[5]SemanticBits, LLC.	[6]Science Application International Corporation (SAIC)
[7]Northern Taiga Ventures, Inc. (NTVI)	[8]Booz Allen Hamilton	[9]NCI - Center for Biomedical Informatics and Information Technology (CBIIT)
[10]Ekagra Software Technologies, Ltd.	[11]Terrapin Systems LLC (TerpSys)	

Other Acknowledgements
GeneConnect – Project - Washington University
GridIMAGE – Project - Ohio State University
caBIO – Project - National Cancer Institute Center for Bioinformatics (NCICB)
caArray – Project - National Cancer Institute Center for Bioinformatics (NCICB)
caTRIP – Project – Duke Comprehensive Cancer Center

Other Acknowledgements
GenePattern – Project – Broad Institute
geWorkbench – Columbia University
Bioconductor – Project – Fred Hutchinson Cancer Research Center

Contacts and Support	
NCICB Application Support	http://ncicbsupport.nci.nih.gov/sw/ Telephone: 301-451-4384 Toll free: 888-478-4423

LISTSERV Facilities Pertinent to caGrid		
LISTSERV	***URL***	***Name***
cagrid_users-l@list.nih.gov	https://list.nih.gov/archives/cagrid_users-l.html	caGrid Users Discussion Forum

Table of Contents

Chapter 1 About This Guide

Purpose

The cancer Biomedical Informatics Grid, or caBIG™, is a voluntary virtual informatics infrastructure that connects data, research tools, scientists, and organizations to leverage their combined strengths and expertise in an open environment with common standards and shared tools. The current test bed architecture of caBIG™, is dubbed caGrid. The software embodiment and corresponding documentation of this architecture constitute the caGrid 1.1 release.

This User Guide addresses caGrid from the perspective of three user roles: service developer, client application developer, and service administrator.

Release Schedule

This guide has been updated for the caGrid 1.1 release. It may be updated between releases if errors or omissions are found. The current document refers to the 1.1 version of caGrid, released in September 2007 by caBIG.

Audience

The primary audience of this guide is the caGrid service developer, client application developer, and service administrator. For additional information about installing and using caGrid, see the caGrid 1.1 Programmer's Guide.

This guide assumes that you are familiar with the java programming language and/or other programming languages, database concepts, and the Internet. If you intend to use caGrid resources in software applications, it assumes that you have experience with building and using complex data systems.

Getting Help

NCICB Application Support

http://ncicbsupport.nci.nih.gov/sw/

Telephone: 301-451-4384

Toll free: 888-478-4423

Using This Guide

This guide is divided into sections depending on the context of the user. The following list briefly describes the contents of each chapter.

- Chapter 1 , this chapter, provides an overview of the guide.

- Chapter 2 describes the three primary caGrid roles for which this guide is written.
- Chapter 3 provides an overview and examples using the Introduce toolkit for service development.
- Chapter 4 describes how to create caGrid data services.
- Chapter 5 introduces the client applications for caGrid services.
- Chapter 6 describes the caGrid security infrastructure, which provides services and tools to administer and enforce security policy.
- Chapter 7 describes the caGrid implementation of a workflow, which provides a grid service for submitting and running workflows that are composed of other grid services.
- Appendix A provides references relevant to caGrid.

Document Text Conventions

The following table shows how text conventions are represented in this guide. The various typefaces differentiate between regular text and menu commands, keyboard keys, and text that you type.

Convention	Description	Example
Bold & Capitalized Command **Capitalized command > Capitalized command**	Indicates a Menu command Indicates Sequential Menu commands	**Admin** > **Refresh**
TEXT IN SMALL CAPS	Keyboard key that you press	Press ENTER
TEXT IN SMALL CAPS + TEXT IN SMALL CAPS	Keyboard keys that you press simultaneously	Press SHIFT + CTRL and then release both.
`Special typestyle`	Used for filenames, directory names, commands, file listings, source code examples and anything that would appear in a Java program, such as methods, variables, and classes.	`URL_definition ::= url_string`
Boldface type	Options that you select in dialog boxes or drop-down menus. Buttons or icons that you click.	In the Open dialog box, select the file and click the **Open** button.
Italics	Used to reference text that you type.	Enter *antrun.*

Convention	Description	Example
Note:	Highlights a concept of particular interest	**Note:** This concept is used throughout the installation manual.
Hyperlink	Links text to another part of the document or to a URL	Overview

Table 1-1 Document Conventions

Chapter 2 Overview of caGrid User Roles

This chapter addresses caGrid from the perspective of three user roles: the Service Developer, the Client Application Developer, and the Service Administrator. Topics for each of these roles are then described in separate chapters in this guide.

Topics in this chapter include:

- Overview on this page
- Relevant Documents on this page
- User Role Definitions on this page

Overview

This guide is intended to provide a user-oriented overview of how various activities can be accomplished using the caGrid software distribution. Some common roles caGrid users assume are described in the following sections. The rest of this guide's content describes how various activities required of these roles can be accomplished with the software. While some of the content provides specific examples and step by step information, it should not be used as a stand alone "tutorial" for caGrid. Additional accompanying documentation is listed below.

Relevant Documents

This User's Guide addresses caGrid from the perspective of three user roles. Additional information about caGrid architecture, design, application programming interfaces (APIs) and API examples, and tool-specific guides can be found in:

Document	Location
caGrid 1.1 Programmer's Guide	http://gforge.nci.nih.gov/frs/?group_id=25
caGrid 1.1 Design Documents and Tool-specific Guides	https://gforge.nci.nih.gov/plugins/scmcvs/cvsweb.php/cagrid-1-0/Documentation/docs/?cvsroot=cagrid-1-0

User Role Definitions

caGrid is primarily an infrastructure or middleware, providing services, APIs, and toolkits for caBIG developers. caGrid provides the common grid infrastructure upon which the Gold compliant grid services and tools are built. While some "end user" tools are provided, the primary consumers of the software are intended to be application or service developers, or service administrators.

Service Developer

caGrid users interested in providing data or analysis routines to caBIG do so by creating and deploying grid services. As such, these users are assuming the role of "Service Developer". The primary tool provided to facilitate the development of grid services in caGrid is Introduce. An overview and examples of using this tool can be found in Chapter 3 Developing caGrid Services. While the aim of this toolkit is to facilitate the creation of caBIG compliant grid services, its use is neither sufficient nor necessary for compliance. Introduce makes the service creation process straightforward, and automatically takes care of most of the caBIG service requirements, but users should still expect to undergo a compatibility review before claiming caBIG compatibility. Service developers may also choose to develop services without making use of Introduce, but they should be sure to meet all service requirements and specifications.

The Service Developer role can be subdivided into two roles, depending on the type of functionality the user is trying to "grid enable." caGrid makes the distinction, in terms of tooling provided and requirements, between Data Services and Analytical Services. Both require that the data types being consumed or produced by the service meet certain requirements, such as being registered in the caDSR and GME. An Analytical Service is generally any service that meets the basic service requirements, and is not a data providing service. Services providing data resources to the grid are required to be developed as Data Services, which in addition to meeting basic service requirements, must implement a standard query operation and language, and expose standardized data service metadata. Service Developers creating Data Services are referred to as Data Service Developers. Service Developers creating Analytical Services are referred to as Analytical Service Developers. It is possible for a Data Service to also provide additional capabilities or operations, so some Service Developers may assume both roles when creating their service.

Typical activities of the Service Developer role include: creation of a grid service, modification or customization of a grid service, and the deployment of a grid service. Service Developers are the primary "producers" of content in the grid of caBIG (in the respect that they provide access to the information or capability, not that they necessarily produce the content itself).

Analytical Service Developer

As described above, Service Developers wanting to create grid services that aim to provide some analytical routine or other business logic are referred to as Analytical Service Developers.

Details about the caGrid support for Analytical Service Developers can primarily be found in the first section of Chapter 3 of this guide.

Data Service Developer

Service Developers wanting to create grid services that allow access to existing data providing resources, such as a caBIG Silver compliant data system, are referred to as Data Service Developers. A Data Service Developer's primary responsibility is to provide clients query access to the underlying data, using a standard query language. These developers may also wish to provide additional capabilities in their service, such as read or update capabilities, and additional more specialized means of query.

Details about the caGrid support for Data Service Developers can primarily be found in the second section of Chapter 3 of this guide, but developers wanting to provide additional capabilities (beyond query) in their service may be interested in the entire section.

Client Application Developer

The counterpart to a Service Developer is a Client Application Developer, who is the consumer of the content to which the Service Developer provides access. All grid services are accessed by making use of a client API or service interface; the developers responsibility for assembling these APIs or interfaces into meaningful applications (or other frameworks) are referred to as

Client Application Developers.

caGrid provides a plethora of tools and APIs for Client Application Developers, and this document is far from an exhaustive list. However, a general overview of the basic concepts of creating client applications and some common examples, are shown in Chapter 5 Developing Client Applications. It is highly recommended that Client Application Developers also peruse the caGrid 1.1 Programmer's Guide to understand the types of functionalities caGrid makes available to them.

Service Administrator

The final type of role caGrid users may assume is that of Service Administrator. caGrid is composed of a number of complex core services that require proper administration. These core services also provide a great deal of capability for Service Developers to configure services to integrate with existing systems for performing such functions as authentication and authorization. The individuals responsible for the proper management and configuration of these services are referred to as Service Administrators. Overviews, step by step examples, and configuration examples for such activities can be found in Chapter 5.

Chapter 3 Developing caGrid Services

This chapter provides an overview and examples using the Introduce toolkit for service development.

Topics in this chapter include:

Recommended Reading

Before building grid services, one should be familiar with basic grid service architecture and also stateful grid services when planning to create asynchronous services and/or stateful services. For more information about grid service architecture and grid middleware, please see the following:

- Overview Papers on Grid Computing
- Globus Best Practices

Overview

The Introduce toolkit is designed to support the three main steps of service development (Figure 3-1):

1) Creating a Basic Service Structure. The service developer describes at the highest level some basic attributes about the service such as service name and service namespace. Once the user has set these basic service configuration properties, Introduce creates the basic service implementation, to which the developer can then add application-specific methods and security options through the service modification steps.

2) Service Modification. The modification step allows the developer to add, remove, and modify

service methods, properties, resources, service contexts, and service/method level security. In this step, the developer can create a strongly-typed service interface using well-defined, published schemas, which are registered in a system like the Mobius GME, as the type definitions of the input and output parameters of the service methods. Once the operations are added to the service the developer can add the logic which implements the methods.

3) Deployment. The developer can deploy the service which has been created with Introduce to a Grid service container (e.g., a Globus or Tomcat service container). A service developer can access the functions required to execute these three steps through the Graphical Development Environment (GDE) of Introduce. The runtime support behind the GDE functionality is provided by the Introduce engine, which consists of the Service Creator, Service Synchronizer, and Service Deployer components. The toolkit provides an extension framework that allows Introduce to be customized and extended for custom service types and discovery of custom data types. The following sections describe the software prerequisites, the Introduce Graphical Development Environment, the Introduce Engine, and the Introduce Extension Framework in greater detail.

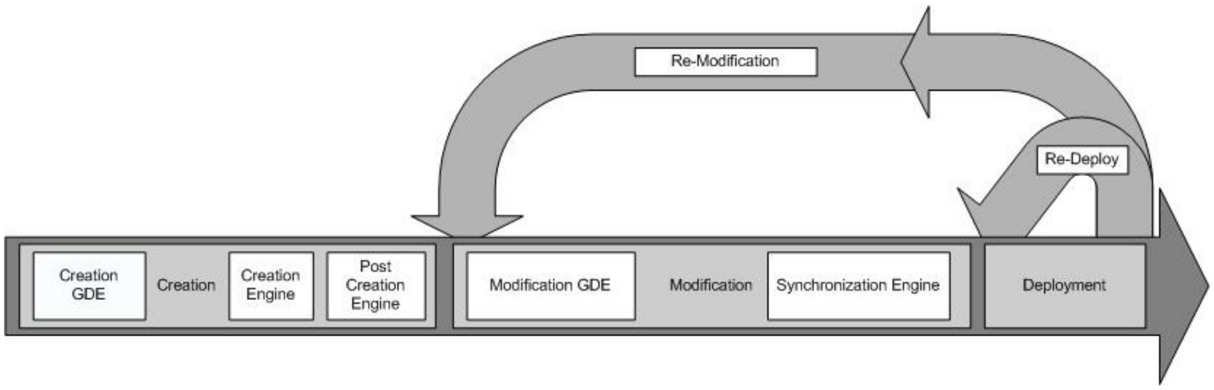


Figure 3-1 Introduce Overall Service Creation Process

Changes from Introduce 1.0

The following is a summary of changes from Introduce 1.0 to the current version, Introduce 1.1.

- Added a new target in the service build file which enables fixing the SOAP bindings to work with custom serialization.
- Added property "perform.index.service.registration" to the deploy.properties to enable or disable index service registration.
- Added two new properties in the deploy.properties enabling the metadata pub/sub timing to be controlled.
    - index.service.registration.refresh_seconds=600
    - index.service.index.refresh_milliseconds=30000
- Added "dev" build.xml and build-deploy.xml which the user can freely edit without expecting changes by updates.

- Moved ant-contrib.jar to ext/lib in the service instead of lib because the service itself does not need that jar as it is only used for building.
- Created a SOAP binding fix that will enable using custom serialization of "some" types of a schema and added it to the services build process.
- Descriptions on the service, methods, inputs, outputs, and faults are now used to create javadoc on the "common" service interface.
- Moved to Policy Decision Point (PDP/authz) based authorization.
    - made the authorization class a PDP
    - configures the service security descriptor to use the PDP
    - added a GUI panel to enable entering a custom PDP chain if desired
- Added support for resource factory/creation code to automatically be generated and placed into any method which is creating and returning a handle to a new service/resource instance.
- Added upgrader framework to enable migration of services from 1.0 to 1.1.
- Added updater framework enabling Introduce to find and install extensions and newer versions directly from the software.
- Refactored service resource framework in order to enable registration of resource properties as well as enable the resource class itself to be user modified and extended as desired. In doing so, the BaseResource class was also renamed to <serviceName>Resource to make it more clear to the developer. This also caused the renaming of the etc/registration.xml to etc/<serviceName>_registration.xml in order to support registration for the all of the resource framework types.
- Only services which have resource properties that are to be registered will attempt to register with the index service.
- Most fields in the GDE which require user input are all dynamically validated and provide graphical feedback to the user if a warning or error is detected.

External Middleware Systems and Tools

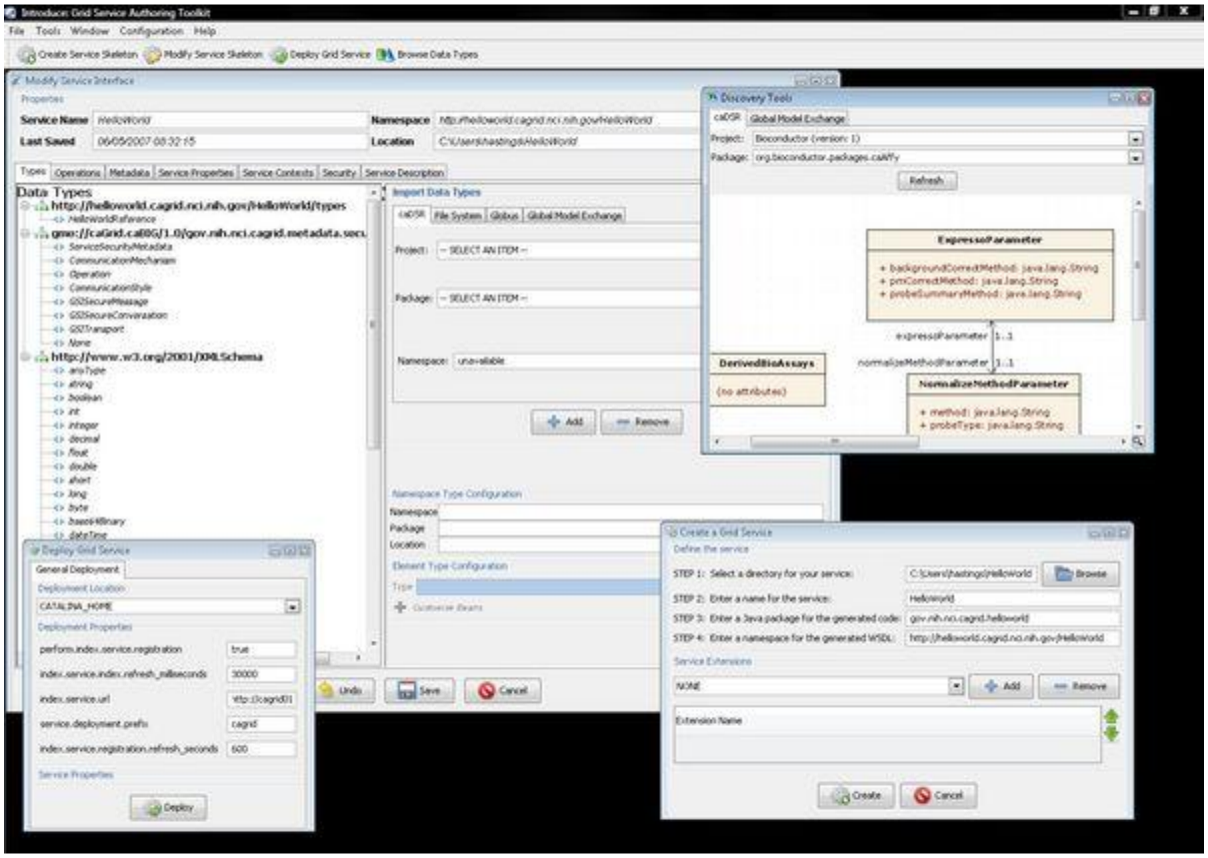
Introduce assumes the availability of two external components to implement its functions: 1) a Grid runtime environment that provides support for compiling, advertising, and deploying Grid services; 2) a repository of data types that is accessible locally or remotely so that the toolkit can pull the common data types for strongly-typed services. The current implementation of Introduce leverages several open-source, middleware systems such as the Globus Toolkit (GT), Apache Axis, and the Mobius Global Model Exchange (GME) service. However, it is designed as a modular system, in which the specific implementations of external components can be replaced.

The GT and Apache Axis implement the core Grid/Web Service support. The Introduce toolkit uses the GT as the underlying Grid runtime environment. It generates the service layout, the service description (WSDL), and the service files such that they are compliant with the GT and Axis and can be readily deployed.

Introduce uses the Mobius GME service as a repository of XML schemas to support the development of strongly-typed services; it also has the ability for custom data type discovery plug-ins, which are described later. The GME is one of the core services provided in the Mobius framework. It implements support for coordinated management of XML schemas in a distributed environment. In the GME, all schemas are registered under namespaces and can be versioned. The concept of versioning schemas is formalized in the GME; any change to a schema is reflected as either a new version of the schema or a totally new schema under a different name or namespace. In this way, data types can evolve while allowing clients and services to make use of old versions of the data type, if necessary. Namespaces enable distributed and hierarchical management of schemas; two groups can create and manage schemas under different namespaces without worrying about affecting each other's services, clients, and programs. In the Grid environment, data elements and objects are exchanged as XML documents conforming to a schema. In such a setting, the schema describes the structure of a simple or complex data element (data object) that is consumed or produced by a service and is exchanged between two endpoints in the environment.

Introduce Graphical Development Environment

The Introduce Graphical Development Environment (GDE) can be used to create, modify, and deploy a grid service (Figure 3-2). It is designed to be simple to use, enable the use of community accepted data types, and provide easy configuration of service metadata, operations, resources, and security. It also allows customized plug-ins to be added for such tasks as discovering data types from grid repositories and for creating custom service style design templates.



Service Name: HelloWorld
Namespace: http://helloworld.cagrid.nci.nih.gov/HelloWorld
Last Saved: 06/05/2007 08:32:15
Location: C:\Users\hastings\HelloWorld

Data Types
http://helloworld.cagrid.nci.nih.gov/HelloWorld/types
HelloWorldReference
gme://caGrid.caBIG/1.0/gov.nih.nci.cagrid.metadata.secu
ServiceSecurityMetadata
CommunicationMechanism
Operation
CommunicationStyle
GSISecureMessage
GSISecureConversation
GSITransport
None
http://www.w3.org/2001/XMLSchema
anyType
string
boolean
int
integer
decimal
float
double
short
long
byte
base64Binary
dateTime

Project: -- SELECT AN ITEM --
Package: -- SELECT AN ITEM --
Namespace: unavailable

Project: Bioconductor (version: 1)
Package: org.bioconductor.packages.caAffy

+ backgroundCorrectMethod: java.lang.String
+ pmCorrectMethod: java.lang.String
+ probeSummaryMethod: java.lang.String
expressoParameter 1..1
normalizeMethodParameter 1..1
(no attributes)
+ method: java.lang.String
+ probeType: java.lang.String

Deployment Location: CATALINA_HOME
Deployment Properties
perform.index.service.registration: true
index.service.index.refresh_milliseconds: 30000
index.service.url: http://cagrid01
service.deployment.prefix: cagrid
index.service.registration.refresh_seconds: 600

Define the service
STEP 1: Select a directory for your service: C:\Users\hastings\HelloWorld
STEP 2: Enter a name for the service: HelloWorld
STEP 3: Enter a Java package for the generated code: gov.nih.nci.cagrid.helloworld
STEP 4: Enter a namespace for the generated WSDL: http://helloworld.cagrid.nci.nih.gov/HelloWorld
Service Extensions: NONE
Extension Name

Figure 3-3 The Introduce Graphical Development Environment (GDE)

The Introduce GDE contains several screens and options for the service developer to 1) create a new service, 2) modify an existing service, 3) discover and use published data types in order to create strongly-typed service methods, and 4) deploy the service.

Service Creation

The service creation component, shown Figure 3-4, enables the developer to create a new grid service. Using the creation interface, the service developer can provide basic information about the service such as:

- Creation Directory

  The creation directory is the location of which the grid service will be generated.

- Service Name

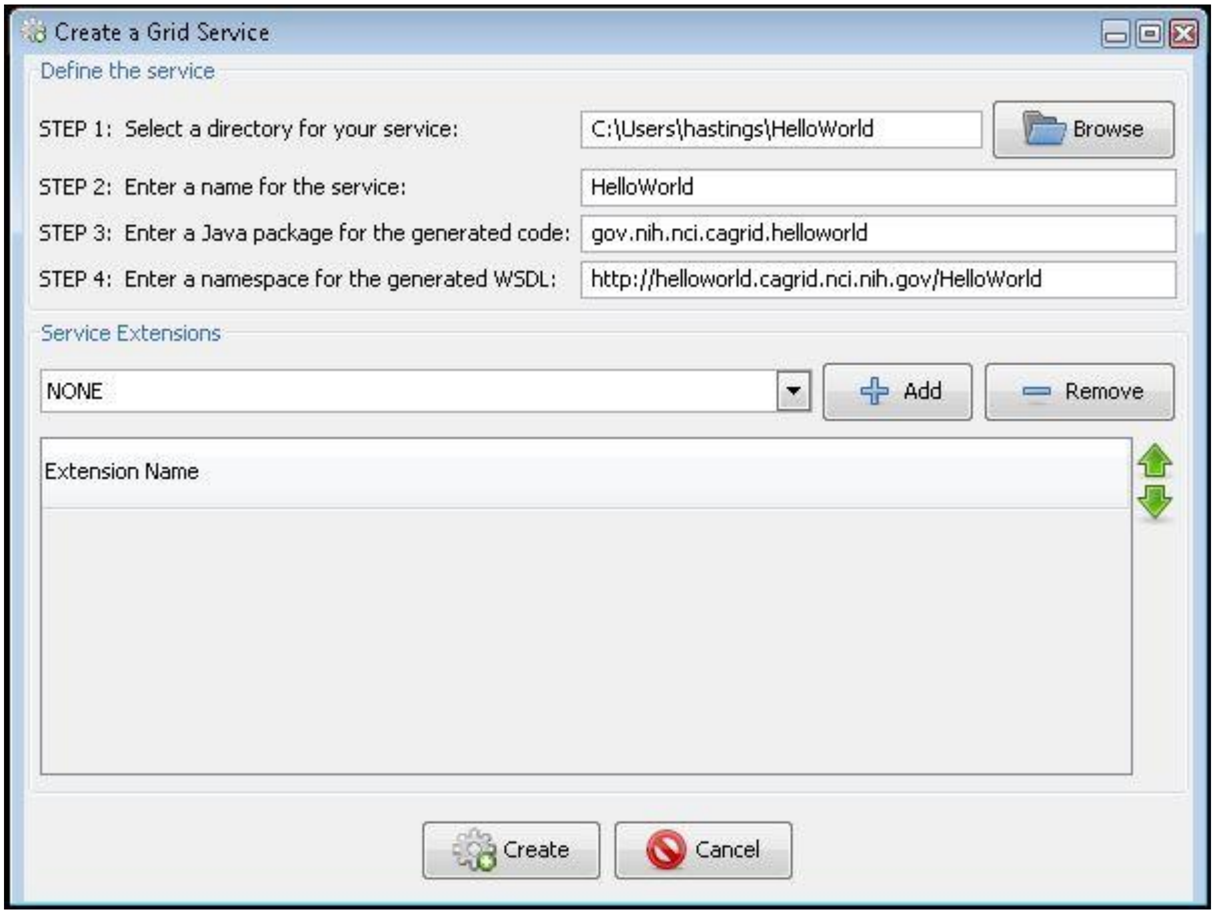
  Service name is the name that will be used to generate the service. The service name must be a valid java identifier.

- Package Name

  The package name is the base package to be used when generated the grid service source code.

- Namespace

  The namespace is the namespace to be used when defining the WSDL of the service.



Create a Grid Service

Define the service

STEP 1: Select a directory for your service: C:\Users\hastings\HelloWorld

STEP 2: Enter a name for the service: HelloWorld

STEP 3: Enter a Java package for the generated code: gov.nih.nci.cagrid.helloworld

STEP 4: Enter a namespace for the generated WSDL: http://helloworld.cagrid.nci.nih.gov/HelloWorld

Service Extensions

NONE

Extension Name

Figure 3-4 Introduce GDE Service Creation Component

The developer also has the ability to add service extensions, which is an Introduce plug-in (see Service Extensions) designed to add customizations to the service. For example, service extensions might add pre-defined operations, resources/resource properties, or security settings. They enable the development of custom service types with predefined methods, which must be implemented. They also enable Introduce to run the custom code implemented in the plug-in, which makes modifications to the underlying service being created. This capability allows the specialization of Introduce to support domain specific common scenarios, further abstracting the individual service developer from responsibilities related to the deployment of grid technologies in a production environment. Once the information has been entered and extensions, if any, have been selected, click **Create**. The Introduce creation engine begins generating the service. After the service is generated it is compiled and the Modification component is displayed.

caBIG

Introduce, when delivered with caGrid, has a button for creating a caGrid service. A custom screen opens for creating either an analytical service or a data service. This screen is similar to the general screen creation described in the previous section, except that in this screen, Introduce extensions are added to the service automatically. By using this screen to create a caGrid compatible service, a series of extensions are added to the service to support caGrid metadata, security, and data service components, if necessary. If a data service is created, a series of screens aids in describing the data service to create.

Service Modification

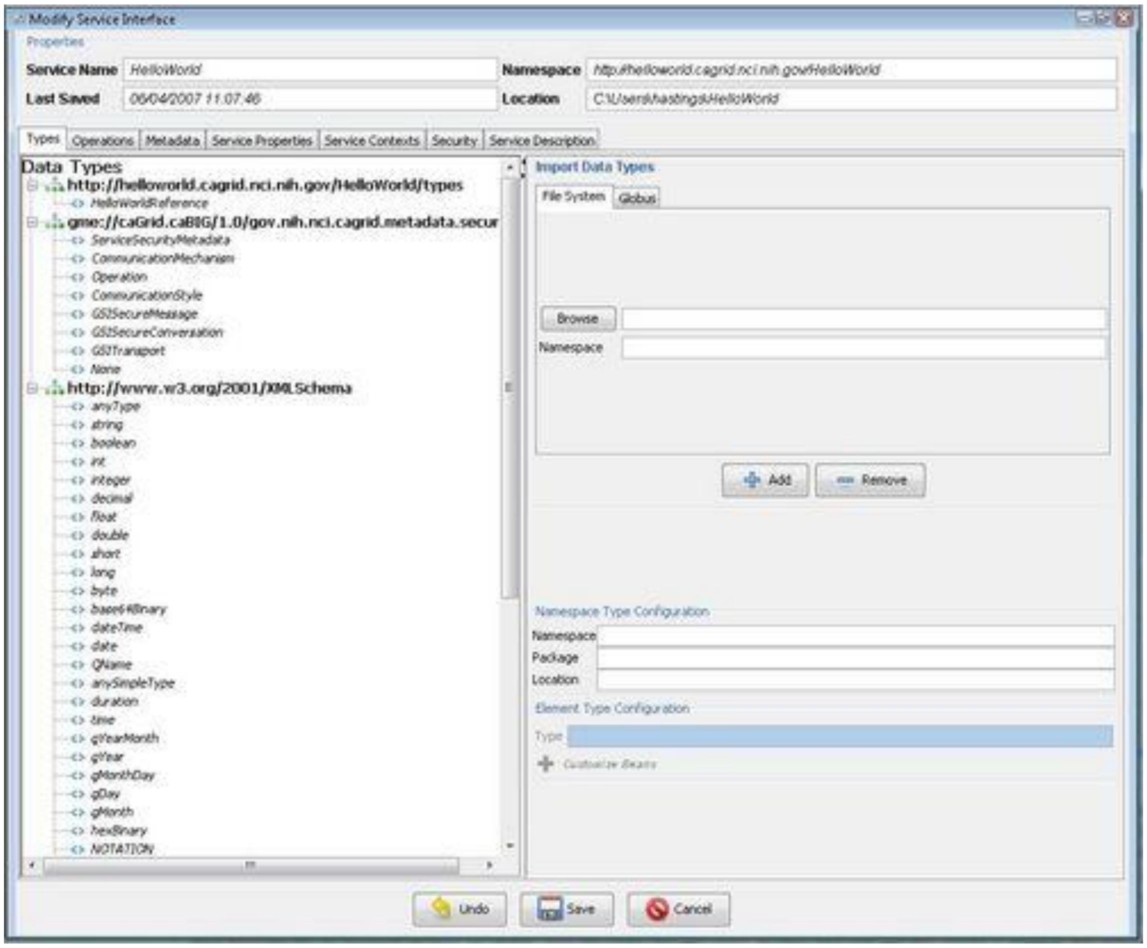
Service modification can be performed on any new or previously modified Introduce generated service. The service developer can perform a series of operations in order to begin to customize the grid service or modify the existing grid service. The overall flow in the modification of a grid service is to first use the namespaces tab to be sure that all the data types that are desired to be used I the grid service have been selected and added to the service. Next the service can choose to either add/remove or modify operations, metadata in the form of resource properties, service properties, security setting, and service contexts.

The following sections describe in more detail how each of the components of the modification viewer can be used to modify the grid service to achieve desired functionality. By selecting **Modify Service** from the main menu, a prompt displays to allow choosing the service to be modified. Once the desired directory containing the service to be modified is selected, the Modification Viewer launches from where the following six types of modifications can be performed on the main service:

- Data Types

Data Types

The first task in the modification of a grid service is to discover the data types that are desired to be used as the input and output types of methods of the service and the data types for describing the resource properties of the service. Adding a data type to the service is equivalent to copying schemas into the schema location of the service and importing the schemas into the WSDL file so that the types can be referenced by the service. This is done via the Types tab of the Graphical Service Modification Environment (Figure 3-5). This tab shows the current types the service is using, and provides access to the data type discovery components (such as the Mobius GME), for selecting and configuring additional types. The “Select Type” frame enables several types of ways to locate data types and bring them into the service. Currently, three main discovery mechanisms (GME, Globus, and File System) are included with Introduce. However, this is extensible via the Discovery Extension described in the Extensions section. Once a set of data types from a namespace are brought into the service, the user has the ability to describe how these data types will be mapped into their respective Java classes. This can, by default, be done automatically by Introduce via Axis. By default, Axis creates new java beans for each data type and also provides a serializer and deserializer for those objects. For example, if a set of objects already exists for a particular data type, then a developer can provide custom classes and serialization/deserialization factories.



Service Name: HelloWorld
Namespace: http://helloworld.cagrid.nci.nih.gov/HelloWorld
Last Saved: 06/04/2007 11:07:46
Location: C:\Users\hastings\HelloWorld

Figure 3-5 Introduce GDE Service Modification Component

Importing Data Types

Using the Graphical Development Environment (GDE), developers can obtain the data types that they want to use for operation input parameters and return types from any data type discovery plug-in. Utilizing common and standard data types, which are defined outside of any application-specific service, enable the creation of strongly typed grid service interfaces. This increases service-to-service interoperability. Once a data type is chosen through the GDE, the data type definition is retrieved, written into the *schema/<service_name>* location of the service, and imported for use in the service WSDL description so that Java beans can be generated and the data types can be programmatically used.

The Introduce toolkit comes with a set of pre-installed discovery plug-ins, such as the Mobius GME and a basic file system browser, which can be used to locate and import schemas. The GME plug-in enables developers to browse, upload, and download schemas published in a GME. These schemas represent the valid data types which can be used during service creation. Using the GME plug-in, a developer can take a schema, create an editable view of the schema, and then submit the schema to the GME. If the namespace of the schema is not managed by the GME to which the schema is submitted, the plug-in will attempt to add the namespace to the GME before submitting the schema. Once the schema has been uploaded, it can be used by anyone in the Grid via the Introduce toolkit. The GME plug-in browser window enables browsing through all the GME published types by namespace and schema name. It presents the user a quick view of the schema and the option to download the schema bundle. The schema bundle contains the schema and all other schemas which are referenced by that schema.

When importing a data type there are several options for acquiring the data type definitions. Introduce is built with the data type definition described in the next section. However, this is a pluggable piece of Introduce. Once a data type is imported using an import tool that data type can be customized for the generation of Java Beans. If a data type is selected on the left side of the panel, the namespace and package name are listed in the lower left panel. This is called the namespace to package map and determines the package name of the Java Beans created for the data type.

Note: The default package name created by Introduce can be changed.

File System Data Type Importing

The File System tab of the Import Data Types Panel is for loading a schemata which contains data types the should service should use from the local file system. Browse to select the schema to import and the click **Add**. The schema and any locally included or imported schemata will be copied to the services schema location in the *schema/<service name>* directory.

Global Model Exchange Data Type Importing

The Global Model Exchange Data Type extension allows browsing for schemata from a remote grid service which is responsible for storing them. Introduce can connect to a GME and load all of the available namespaces for which the GME is storing schemata. Once a namespace is selected, the Name drop-down menu is populated with all of the available schema names for that schema. Click **Add** to download and import the schema and any imported schemata into the service. The **Configuration** menu for the GDE allows changing the location of the GME to use.

Globus Data Types Importing

The Globus Data Types extension allows importing a schema from the Globus toolkit into a service. A Name drop-down menu displays and contains a list of the available schemata from the current installation of the Globus toolkit. Click **Add** to add the schema to the service's available data types list.

Reimporting a Modified Data Type

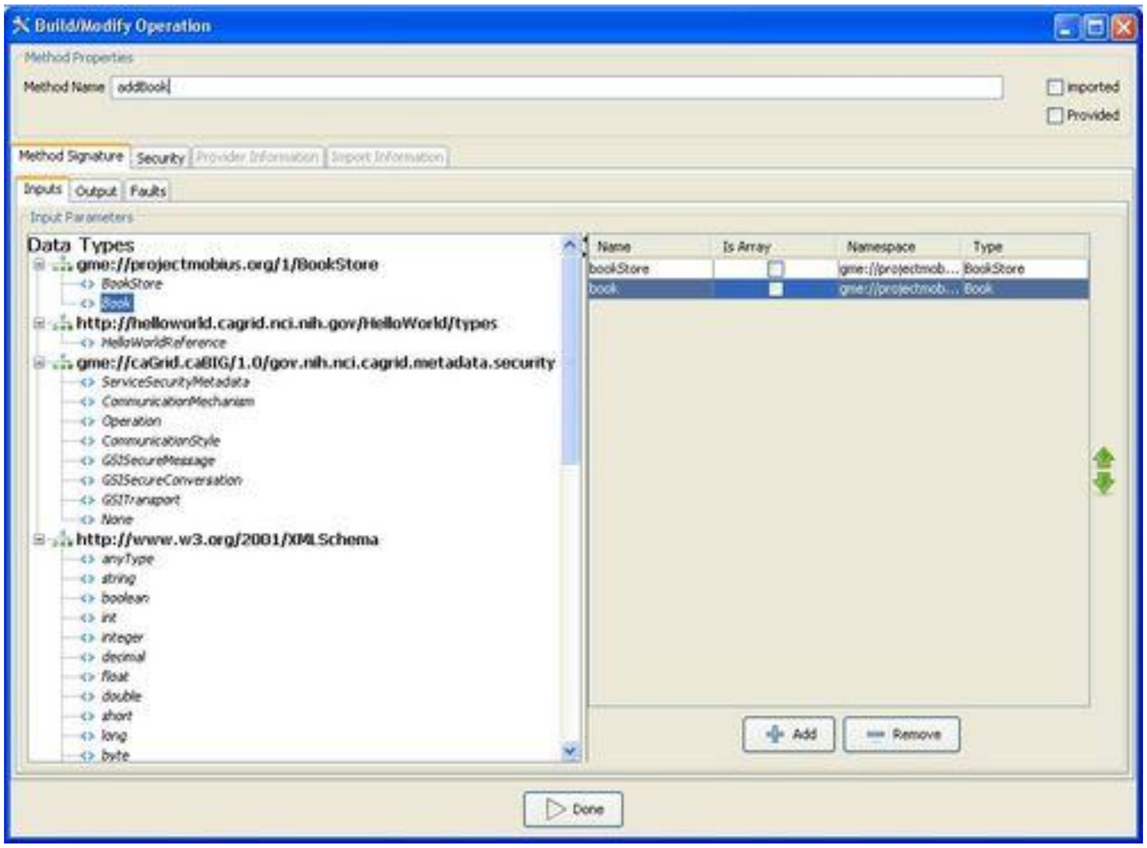
Introduce allows reimporting a data type if a particular schema, which may have been modified or extended, needs to be reimported. In order to do so, ensure the **Namespace Type Replacement Policy** configuration setting in the Introduce **Configuration-->Preferences** menu is set to **warn**. Once this is set, browse back to the data model and import it again.

Using Custom or Preexisting Java Beans

Once a namespace and its corresponding data types have been imported into the service, each data type can be further customized. For a particular data type, a custom java bean that already exists can be selected instead of having Introduce create the java beans for the service by clicking the **+** sign beside the **Customize Bean** label. A customization panel for the selected data type displays. In this panel, enter information for three fields to support a custom java bean: the classname of the bean to be used (ensure the package name entered above matches the package name entered for the custom bean), the deserializer factory class, and the serializer factory class. For more information on using custom serialization or what it means to be a custom bean, please refer to the caGrid wiki (http://www.cagrid.org/mwiki/index.php?title=CaGrid:Serialization).

Operations

The Operations tab of the GDE Service Modification interface allows adding, removing, or modifying operations on the service (Figure 3-6). For each operation, set the input parameters, return type, and any fault types that can be thrown from each service method. The security configuration of the operation should also be set, if desired. The input and output types can be selected from the types tree on the left. This tree represents the available data types which can be used by this service. If any input parameter or output type is to be an array, the array checkbox must be checked in the table on the right. Also, once an input parameter is added, the parameter is assigned a default name, which can be changed by selecting the cell in the name column and editing the text. There are two ways to add faults; 1) select a type from the types tree which extends WSRF BaseFaultType 2) create a new fault which tells Introduce to create a new fault type which extends the BaseFaultType.



Method Name: addBook

```
Data Types
gme://projectmobius.org/1/BookStore
    BookStore
    Book
http://helloworld.cagrid.nci.nih.gov/HelloWorld/types
    HelloWorldReference
gme://caGrid.caBIG/1.0/gov.nih.nci.cagrid.metadata.security
    ServiceSecurityMetadata
    CommunicationMechanism
    Operation
    CommunicationStyle
    GSISecureMessage
    GSISecureConversation
    GSITransport
    None
http://www.w3.org/2001/XMLSchema
    anyType
    string
    boolean
    int
    integer
    decimal
    float
    double
    short
    long
    byte
```

Name	Is Array	Namespace	Type
bookStore	☐	gme://projectmob...	BookStore
book	☐	gme://projectmob...	Book

Figure 3-6 Introduce GDE Method Modification Component

Using a Preexisting Operation Implementation

The implementation of a described operation may already exist in another class which is provided by a jar file. Introduce can be directed not to stub this methods server side implementation but instead call this provided method implementation directly in the class provided. To use this functionality, select the **Provided** checkbox and enter the class name attribute in the **Provider** tab. The class name attribute will point to the fully qualified class name of the class which implements this WSDL described operation. The jar file that contains the provided class which implements this operation must also be copied into the lib directory of the service. This will ensure that the operation will be located at the time the operation calls on the service. For more information on this particular topic, refer to the Globus Documentation on Operation Providers (http://www.globus.org/toolkit/docs/4.0/common/javawscore/developer-index.html).

Importing Operations

Operations can also be imported from other services. Importing an operation allows the service to implement the exact same operation signature, which in turn allows the service to have an operation which has the exact same WSDL signature of the imported operation. This would enable either client to invoke this operation on either service. Importing can be done in two ways: 1) from an Introduce-generated service or 2) from a WSDL file.

From an Introduce-generated service, browse and select the Introduce-generated service which contains the operation to be imported. A list of services which contain this method are available; select the service from which you want to import the operation. The methods signature is imported and a prompt displays to ensure copying the WSDL and XSD files needed to import the method into the schema<service name> directory of the service.

From a WSDL file, obtain the WSDL and its corresponding XSDs in the schema/<service name> directory of the service. Next browse to those WSDL files and select the port type from which to import the operation. Importing a method across services assures not only that each service has protocol-compatible methods but also that each service's method can be invoked by the same base client. This enables the notion of basic inheritance in grid services and is discussed further in Operation Importing.

Resource Properties

Service state information and metadata in the form of resource properties can be added, removed and configured via the Metadata tab of the GDE Service Modification interface. The metadata elements which are added to the service can be populated by a file statically or managed dynamically within the service. Also, these metadata entities can be registered with an index service so that users can use the metadata to locate the service.

Select the Metadata tab to display a list of available data types that can be used for metadata the left panel. The right panel displays the list of currently chosen data types. Double-click on a data type in the left panel to add it to the main service's metadata list. Any of the service's metadata can be initially populated from a file, if desired. Once the service is started in the container, the file will be used to populate the particular metadata object in the service. Each metadata in the service can also be selected for publishing to an index service, which enables some or all of the metadata to be used to locate the service via an index service.

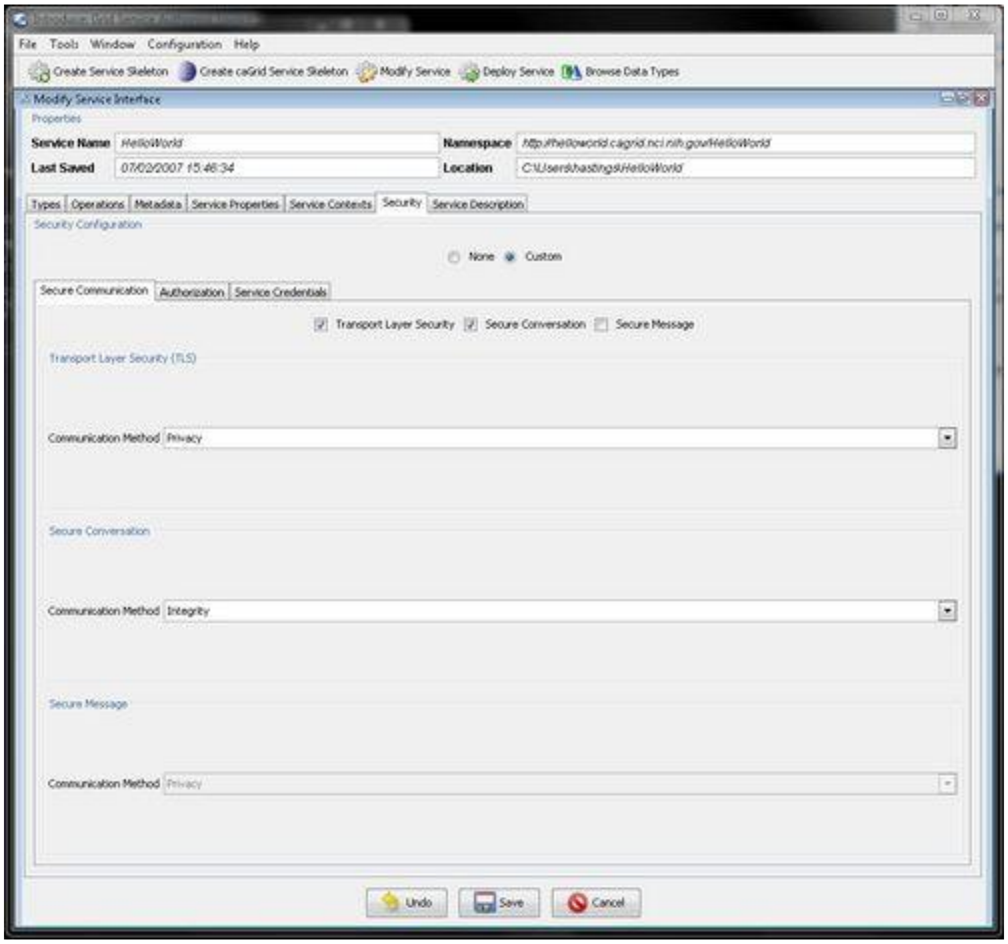
Service Properties

Service properties are key value pairs which can be set at deployment time and are available to the server side implementation of the service at run time. This enables passing in configuration variables to the server side of the service at deployment. These key value pair properties can be declared in "Service Properties" tab of the GDE Service Modification interface. Once the "Service Properties" tab is clicked the main panel will show a table of the service properties. The bottom panel has an entry for which can be used to create a new service property. The properties will be confirmed and/or can be changed from there default values at service deployment time. The variables can then be accessed inside the user's implementation of the operations through the services *ServiceConfiguration* class. For example, if you add a property called *foo* under the service properties tab, and then save the service. Then look at the source code for the <service package>.service.ServiceConfiguration.java class you will note that it now has available methods for *string getFoo()* and *void setFoo(string foo)*. These operations are now available to your service and can be used to pass properties into your service at deployment time as well as other users for configuring and sharing properties in your service. The <service

package>.service.ServiceConfiguration.java contains a static method for obtaining an instance of itself called *getConfiguration()*. Any call to that from anywhere in the service will return the handle to the *ServiceConfiguration* instance and hence access to the service properties.

Security

Introduce exposes the functionality of Globus GSI through a set of panels which enable customizing security for the entire service or specific methods of a service context (Figure 3-7). Any of the GSI configuration scenarios can be selected such as Transport Level Security with Integrity and Secure Communication with Privacy. Introduce also enables configuring a particular service, operation, or resource for authorization. Introduce comes with capabilities to configure authorization using Grid Grouper and/or Common Security Model, or a Custom PDP-based authorization chain. Graphical panels enable describing an authorization policy which must be met in order to provide access to the particular service or operation. For more information about configuration options for Secure Conversation or Secure Credentials, please refer to documentation for the GSI framework.



Properties

Service Name: HelloWorld
Namespace: http://helloworld.cagrid.nci.nih.gov/HelloWorld
Last Saved: 07/02/2007 15:46:34
Location: C:\Users\hastings\HelloWorld

Security Configuration

None [✓ Custom]

[✓ Transport Layer Security] [✓ Secure Conversation] Secure Message

Transport Layer Security (TLS)

Communication Method: Privacy

Secure Conversation

Communication Method: Integrity

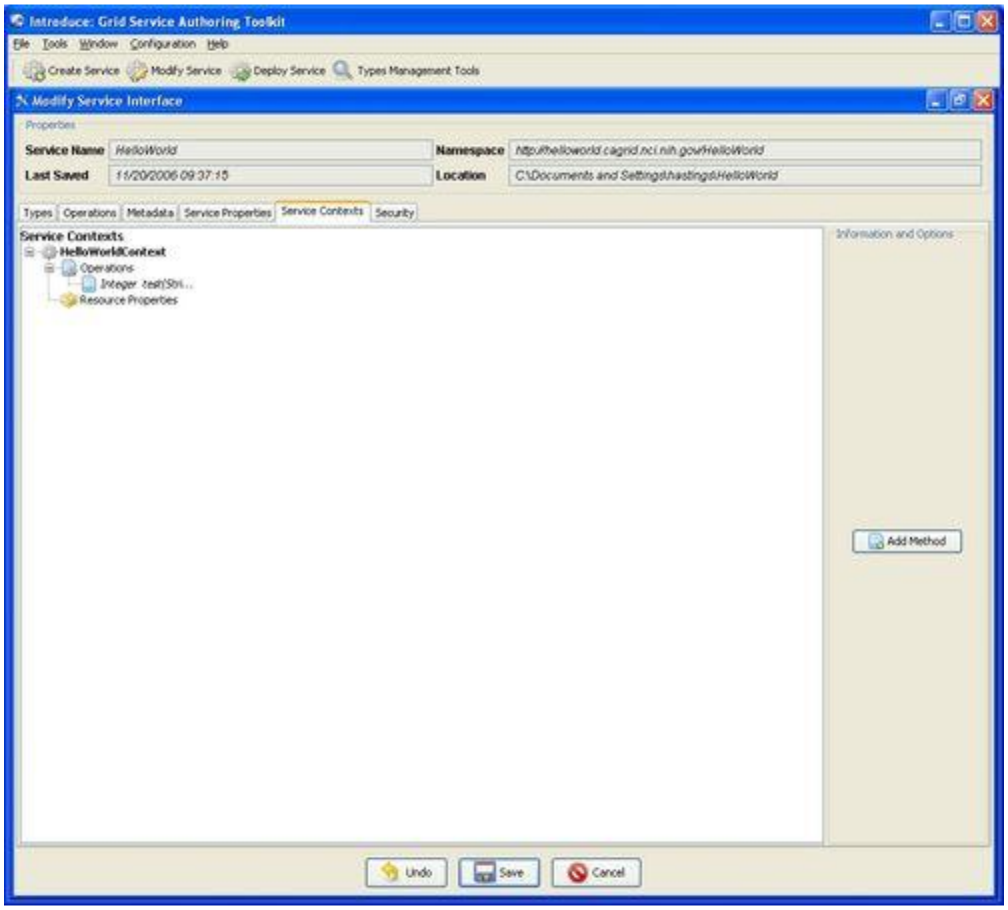
Secure Message

Communication Method: Privacy

Figure 3-7 Introduce GSI Security Configuration Panel

Service Contexts

A advanced feature which can be enabled at modification time is the addition or removal of service contexts. A service context is a sub-service or complimentary service which is used with the main service or some other service context. The service context is comprised of the service, resource, operations, and resource properties. Thus, a service context is exactly the same as the main service, except that it is not a singleton-based resource and instances can be more dynamically created and/or destroyed. Contexts can be added via the **Service Contexts** tab of the GDE Service Modification interface (Figure 3-8).


Introduce: Grid Service Authoring Toolkit
File Tools Window Configuration Help
Create Service Modify Service Deploy Service Types Management Tools
Modify Service Interface
Properties
Types Operations Metadata Service Properties Service Contexts Security
Service Contexts
HelloWorldContext
Operations
Integer test(Stri...
Resource Properties
Information and Options
Add Method
Undo Save Cancel

Service Name: HelloWorld
Namespace: http://helloworld.cagrid.nci.nih.gov/HelloWorld
Last Saved: 11/20/2006 09:37:15
Location: C:\Documents and Settings\hastings\HelloWorld

Figure 3-8 Introduce GDE Service Contexts Modification Component

Service contexts define additional of operations needed to support the desired service functionality. This is enabled by using WSRF capabilities of the Globus Toolkit. As an example, if an operation on the main service enables a database to be queried, that operation might create a resource in another context and return the handle to that context to the user as opposed to the full query result set. This secondary context can then enable the user to iterate through the query results. This is accomplished by operations or resource properties to this secondary service context which will be responsible for iteratively giving results to the user. It should be noted that multiple instances of these contexts can be created and executed concurrently; one for each query that comes in, for example. This style of grid service is supported by the WSRF specifications. Though the details of the WSRF-implementation of these concepts are abstracted away from developers, it is worth noting how they are realized. This is described in detail in other sections. Introduce makes it easier for service developers to create such complex services, via the GDE, without having to fully understand the underlying service implementations. Anything that can be done to the main service, except service properties which are globally accessible, can be added to a service context. For example, resource properties can be added and used to maintain state or to publish metadata to an index service. Also, operations can be added to the service context and can also be implemented in the service itself or in the service's resource if they are acting on the state of the instance of the resource.

A Stateful Grid service is composed of several key components which make it able to maintain state and enable a client to invoke the service several times under the same context. A stateful grid service is composed of the service, a resource home, and the resource type. This service organization can be used in many different scenarios. For example, when an operation on the service is invoked, the service can be implemented to handle that operation. Alternatively, if the operation is addressing a particular resource instance in the service, the service can lookup the resource and call whatever might be necessary to call on the particular addressed resource.

Utilizing the Factory Pattern

In order to create instances of a resource, something must tell the resource home of the service to create them. This is typically done using the factory pattern. A service or method is responsible for telling the resource to create a new instance of a resource type for the service. An example of the factory pattern in action is shown in Figure 3-9.

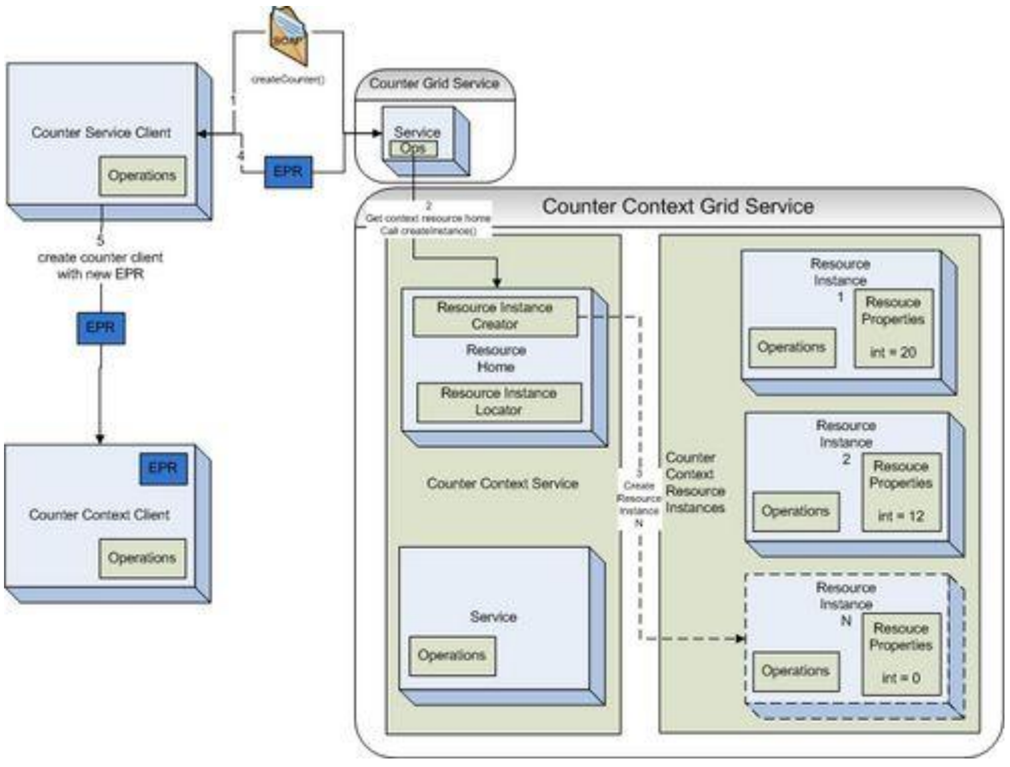


Figure 3-9 Utilizing Grid Service Factory Pattern

A factory service, the Counter Grid Service, contains a create operation which is exposed as a grid service method. When the client invokes this method, the Counter Grid Service locates the resource home of the Counter Context Grid Service and asks it to create a new resource instance. Once this resource instance is created the resource home returns a pointer or address to this resource instance called an EndPointReference (EPR). This EPR is then returned from the create method back to the client so that it has a pointer to the new resource that it can act on.

In order to act on this resource it needs to construct a client that can talk to the grid service representing this resource type (**Error! Reference source not found.**). The Counter Context Client is constructed using the EPR to address the specific resource context for the service. Next, the *incrementCounter()* operation is called on the Counter Context Grid Service. The client makes the call to the grid service and the service looks at the EPR sent by the client. The service takes the EPR and gives it to the resource home and asks for the resource instance back which is represented by the particular EPR. Finally the service runs its logic to increment the counter, which in this case is represented as a resource property of the resource.

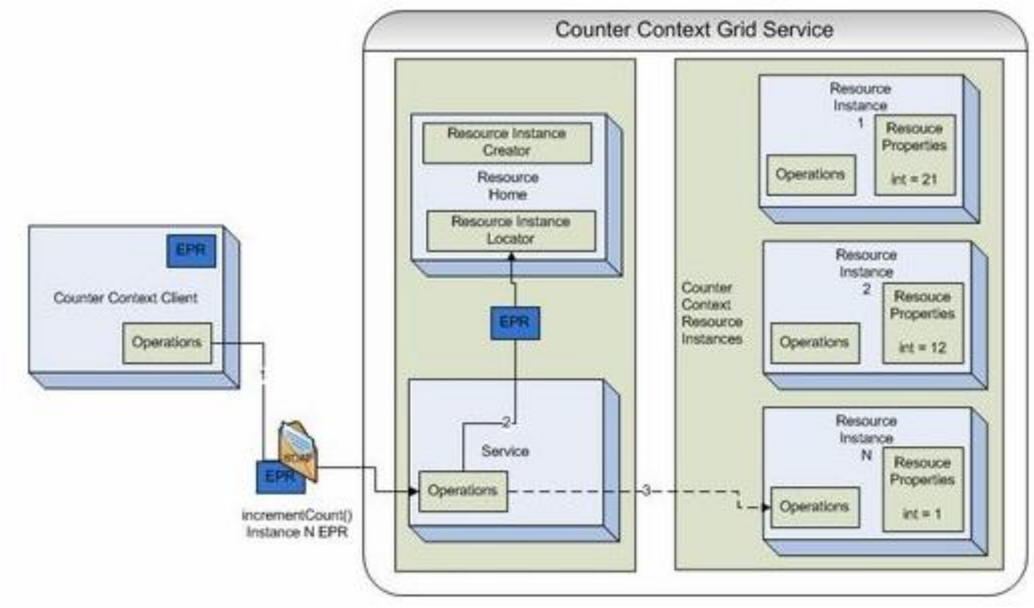


Figure 3-10 Invoking a Stateful Grid Services

An example of the code that is required to implement the *incrementCounter()* method in the CounterContextServiceImpl.java is shown below.

```
package example.counter.context.service;


import java.rmi.RemoteException;


/**
 * TODO:I am the service side implementation class. IMPLEMENT AND DOCUMENT ME
 *
 * @created by Introduce Toolkit version 1.1
 */
public class CounterContextImpl extends CounterContextImplBase {


    public CounterContextImpl() throws RemoteException {
        super();
    }


    public void incrementCounter() throws RemoteException {
        int currentCounter;
        try {
            currentCounter = getResourceHome().getAddressedResource().getIntValue();


            currentCounter++;
            getResourceHome().getAddressedResource().setIntValue(currentCounter);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


}
```

Implementing the Service

When an operation is added in the Introduce GDE and is saved by clicking **Save**, Introduce adds the new stubbed method into the <service package>.service.<service name>Impl.java class. The developer is then responsible for implementing the method prior to deployment. For example, the code snippet below would be generated in the <service package>.service.<service name>Impl.java if the developer added an *add* operation to the service with Introduce that accepted two integers and returned one integer.

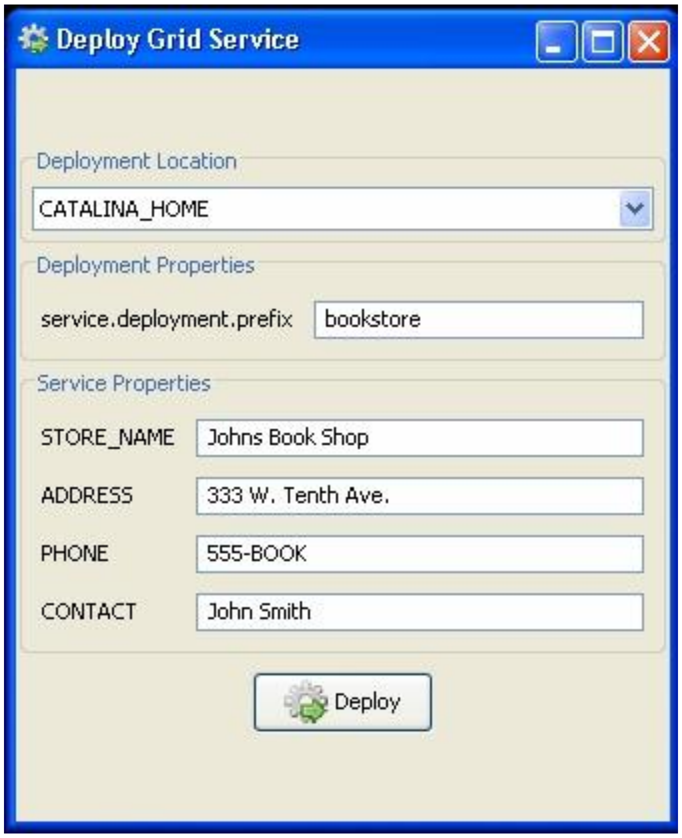
```
public int add(java.math.BigInteger a,java.math.BigInteger b) throws RemoteException {
    //TODO: Implement this autogenerated method
    throw new RemoteException("Not yet implemented");
  }
```

The developer would then have to edit this method to implement the logic the service would execute on invocation.

```
  public int add(java.math.BigInteger a,java.math.BigInteger b) throws RemoteException {
    return a + b;
  }
```

Deployment

The **Deployment** tab of the GDE allows deploying the implemented grid service, which has been created with Introduce, to a Grid service container (Figure 3-11). The toolkit currently supports deploying a service to either a Globus or Tomcat Grid service container. However, support for other deployment options can easily be added to the GDE. The Deployment tab allows populating service configuration properties, which the service will have access to at runtime. Then the service is deployed to the selected container.



Deployment Location

CATALINA_HOME

Deployment Properties

service.deployment.prefix	bookstore

Service Properties

STORE_NAME	Johns Book Shop
ADDRESS	333 W. Tenth Ave.
PHONE	555-BOOK
CONTACT	John Smith

Figure 3-11 Introduce GDE Service Deployment Component

Introduce enables deployment to three container types: Globus, Tomcat, and JBoss. Select the container to deploy to from the **Deployment Location** drop-down menu.

Note: If using Tomcat of JBoss, Globus must first be deployed to the container before any services are deployed. Information on deploying Tomcat to Globus can be found here: http://www.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#javawscore-admin-tomcat-deploying. For more information on how to use JBoss with Globus (supported by caGrid only), please refer to http://www.cagrid.org/mwiki/index.php?title=CaGrid:How-To:DeployGlobusToJBoss

Using the Client

Introduce generates a client API for the service which is exactly as described within the graphical editing environment. The client API can be used in order to leverage this type of service from another application or service. The client API contains four constructors which can be used, each of which is different depending on whether it has a handle or requires the use of an address and security.

```
/**
 * Takes in the url of the service to connect to as a string
 */
HelloWorldClient(String url)
/**
 * Takes in the url of the service to connect to as a string and
 * a proxy to be used to represent the credentials or the caller
 */
HelloWorldClient(String url, GlobusCredential proxy)
/**
 * Takes in the epr which refers to the service or resource
 */
HelloWorldClient(EndpointReferenceType epr)
/**
 * Takes in the epr which refers to the service or resource and
 * a proxy to be used to represent the credentials or the caller
 */
HelloWorldClient(EndpointReferenceType epr, GlobusCredential proxy)
```

Figure 3-12 New client handle to service "HelloWorld"

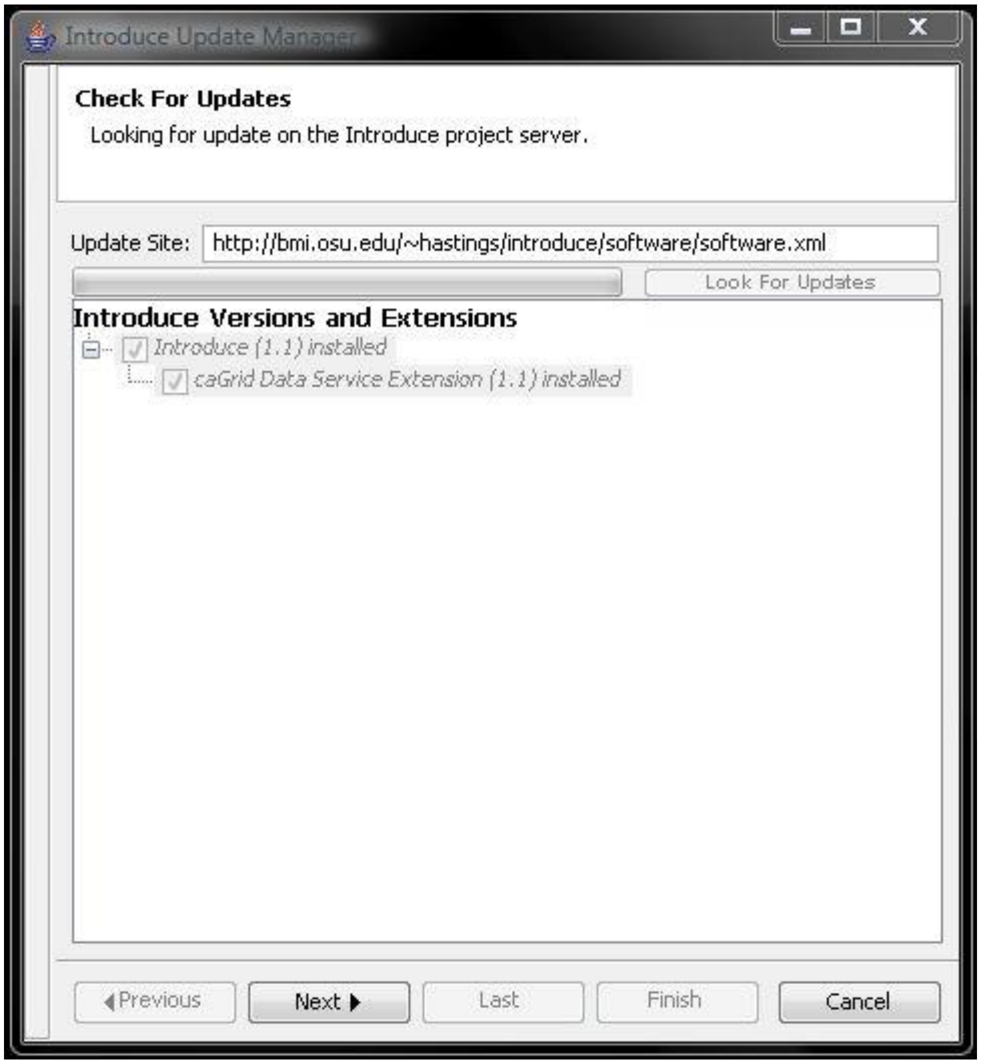
Once a client handle is constructed, each of the operations which were created in the service are available as operations to this newly constructed client instance. The following example contains a snippet of code which creates a new client handle to a service called "HelloWorld" and calls the "echo" operation (Figure 3-13).

```
try {
 HelloWorldClient client = new HelloWorldClient("http://localhost:8080/wsrf/services/HelloWorld");
 client.echo("Testing)";
} catch (Exception e) {
 System.out.println("Problem creating handle to or calling service" + e.getMessage(), e);
}
```

Figure 3-13 Call to "echo" operation

Software Updates

Introduce is now capable of downloading and installing new extensions, upgrades to older extensions, and newer versions of itself. The Help menu in the GDE contains a **Check for Updates** button which opens a wizard that looks for any software updates or new packages to download and install (Figure 3-14).



Looking for update on the Introduce project server.

Update Site: http://bmi.osu.edu/~hastings/introduce/software/software.xml

Introduce Versions and Extensions

- Introduce (1.1) installed
  - caGrid Data Service Extension (1.1) installed

Figure 3-14 Check for Updates wizard in Introduce

Service Migration

Introduce can also help migrate a service to a newer version of Introduce (Figure 3-15). If an attempt is made to open a service generated with an older version of Introduce (1.0 and older), Introduce prompts the user to proceed with the migration process. The migration process is fully automated and when complete, reports what might be left for adjustment based on potentially custom changes or any errors during the process.

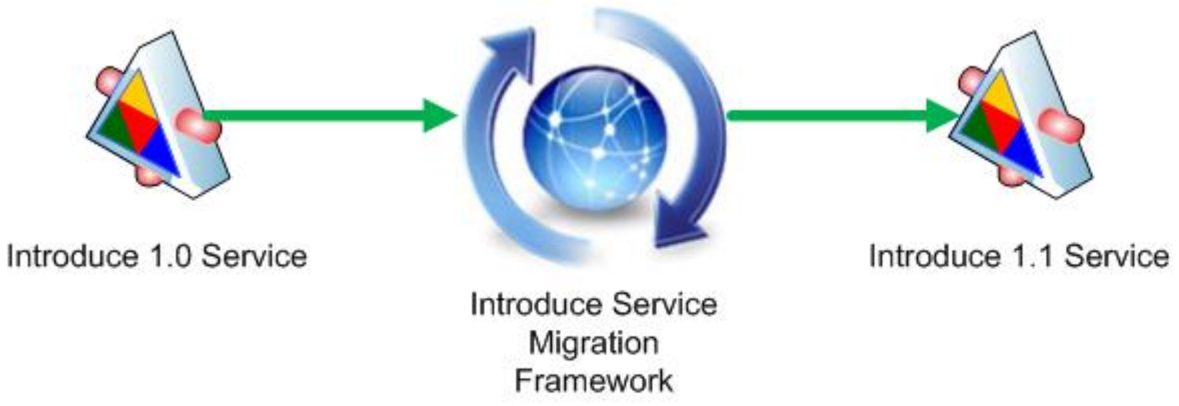


Figure 3-15 Introduce Migration Service

When using Introduce to open a service for modification, it checks the service to see which version of Introduce and its extensions were used to create/modify the service. If those versions are different from those installed in the Introduce being used, it notifies the user that the service needs upgrading. When prompted, a decision must be made to either:

- **Upgrade**: upgrade the service to the version with which Introduce can properly work.
- **Open**: attempt to have Introduce work with it without upgrading, which is potentially dangerous and not recommended.
- **Close**: do nothing to the service and do not proceed with the modification process

When the service is upgraded, a report indicating the major changes and potential issues is displayed for review. Once issues have been addressed from the report, click **Proceed** and the Modification Viewer opens displaying the newly upgraded service. If the report warn that a potential modification is required to finish the Upgrade process, click **Edit**. This stops the upgrade process so changes can be made. If needed, the service can be modified at a later time and Introduce should be able to work with the service. If an upgrade is not desired, click **Roll Back** to restore the service to its previous state.

Chapter 4 Creating caGrid Data Services

This chapter describes a set of extensions to the Introduce Toolkit with which grid service developers can create data services.

Topics in this chapter include:

- Introduction on this page
- User Interface Components and Details on this page
- Functionality of the Extension on page 36

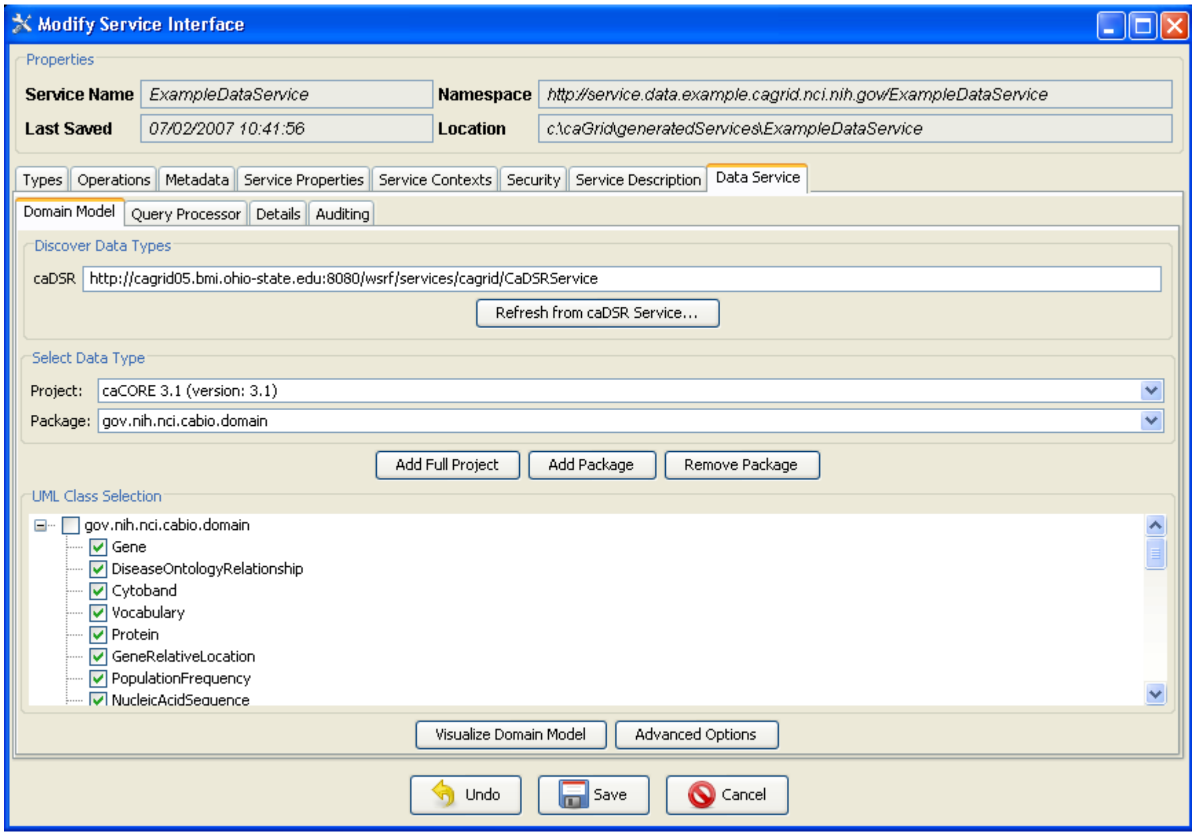
Introduction

caGrid Data Services can be built with a set of extensions to the Introduce Toolkit. This provides grid service developers with a simple and well defined starting point to create caBIG gold compliant Data Services. When the extension has been selected in creating a new service, the user is presented with both a standard service modification interface and a new tab is placed on the interface containing options specific to configuring data services.

User Interface Components and Details

Service Modification Interface

When selected from the caBIG creation dialog in Introduce, the service developer is presented with a dialog allowing selection of a data service style. This dialog also allows the developer to enable the WS-Enumeration or Bulk Data Transfer features of the data service. When the initial creation process is complete, the Service Modification interface displays (Figure 4-1).



Service Name: ExampleDataService
Namespace: http://service.data.example.cagrid.nci.nih.gov/ExampleDataService
Last Saved: 07/02/2007 10:41:56
Location: c:\caGrid\generatedServices\ExampleDataService

caDSR: http://cagrid05.bmi.ohio-state.edu:8080/wsrf/services/cagrid/CaDSRService

Project: caCORE 3.1 (version: 3.1)
Package: gov.nih.nci.cabio.domain

- ☐ gov.nih.nci.cabio.domain
  - ✓ Gene
  - ✓ DiseaseOntologyRelationship
  - ✓ Cytoband
  - ✓ Vocabulary
  - ✓ Protein
  - ✓ GeneRelativeLocation
  - ✓ PopulationFrequency
  - ✓ NucleicAcidSequence

Figure 4-1 Data Services configuration UI

The Data Service extension adds a new graphical component to the Introduce Toolkit's service modification view. This tab contains several sub tabs, each containing major points of configuration for the data service:

- Domain Model
  - Provides facilities for selecting the domain of data types available to be queried by the data service.
- Query Processor
  - Allows selecting an implementation of CQL to use in the data service and provides for configuration of its parameters.
- Details
  - Serializes data types, mapping classes to XML schema element names, and enabling/disabling query validation.
- Auditing
  - Provides for selection and configuration of auditors for various processes of the data service infrastructure.
- Enumeration (Optional)

- If the WS-Enumeration or BDT feature is enabled for data services, this tab will be available to choose a server side implementation of the WS-Enumeration specification. Generally this will not need to be changed from the default implementation.

Domain Model Tab

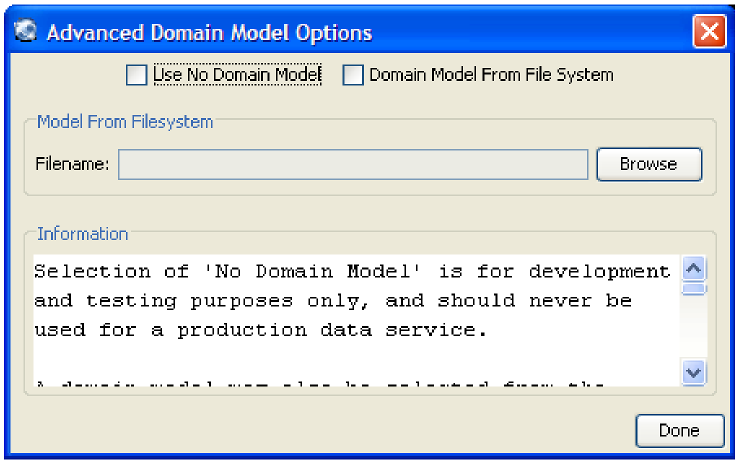
On the Domain Model tab, select the types exposed by this data service. Data types are derived from information stored in the caDSR.

The top field labeled **caDSR** indicates the URL of the caDSR grid service to access. The button **Refresh from caDSR Service** queries the grid service to retrieve a list of Projects and UML Packages available in the caDSR. Projects and packages can be selected with the two drop down menus provided. The **Add Full Project** button inserts all packages and classes from the project into the domain model. The **Add Package** button inserts a single selected package and the **Remove Package** takes a package out of the domain model. When a package is added, its contents display in the tree below the buttons. This tree contains checkboxes next to the package and the individual classes. Placing a check next to a class adds it to the domain model as an object which can be queried, and by default, used as a type which can be targeted with a query. Placing a check by a package automatically adds all the classes in that package in the same way.

Each package selected to participate in the domain model must be mapped to an XML schema for serialization across the grid. If a package is selected for which no schema can be found, the data service extension prompt the service developer to locate it.

The domain model may be displayed graphically by clicking the **Visualize Domain Model** button. Note that this will cause the domain model information to be retrieved from the caDSR service before it can be displayed, which can be a time consuming process. A dialog with a progress bar keeps the developer apprised of the progress of this process.

The **Advanced Options** button opens a dialog enabling the service developer to control advanced options for domain model selection (Figure 4-2).



```
Selection of 'No Domain Model' is for development
and testing purposes only, and should never be
used for a production data service.
```

Figure 4-2 Advanced Domain Model Options

The **No Domain Model** check box disables use of a domain model in the data service. Selecting this option is intended for use when testing other aspects of a service. The option to use a domain model from the file system is also intended for testing purposes. Since building a domain model from the caDSR can be a time consuming process, the domain model may be generated from the caDSR elsewhere and specified here. Selecting either of these options will disable selecting the domain model from the caDSR on the Domain Model configuration tab.

Query Processor Tab

The Query Processor tab shows a drop down of all currently available CQL Query Processor implementations. If a query processor other than the one displayed is required, the jar file containing it may be added to the service by clicking **Add Jar**. When a new jar file is added to the service, both the list of jars and the query processor drop down update to reflect the changes. The added jar files are copied into the service's library directory, and deployed with the service. Selecting a class from the drop down makes it the query processor implementation which is invoked by the data service to handle queries at runtime. Its configuration properties (if any) are shown in the table at the bottom of the screen. This table shows the name of every parameter, its default value, and an editable field where a custom value may be entered. These parameters are stored as service properties in the generated service and are made available at runtime through JNDI. Query processors may optionally supply their own configuration user interface, which can be displayed by clicking **Launch Query Processor Configuration**.

Custom CQL Query processors may be implemented to support querying over a specialized data source by extending the provided base class *gov.nih.nci.cagrid.data.cql.CQLQueryProcessor.*

Details Tab

The Details tab allows configuration of lower level functionality in the data service. This tab contains a table which allows configuring the way each type in the domain model is serialized by the data service, as well as a flag to indicate if each type may be targeted and returned by a CQL query. Types can be selected either individually or in groups and their serialization is configured by a popup menu. This menu allows selecting either the default serialization, SDK serialization for caCORE SDK generated objects, or a custom serialization. Selecting SDK serialization assigns the SDK serializer and deserializer factories to the types. Custom serialization presents the user with a dialog box in which to enter the serializer and deserializer classes which will be assigned to the schema type. This information is recorded in the generated client-config.wsdd file, as well as the server-config.wsdd. These configuration files may be used later in the data service clients and utilities to properly deserialize results of queries. This tab also allows selecting what type of query validation to be performed. Selecting to **validate CQL syntax** causes the Data Service to ensure that the submitted CQL query conforms to the CQL schema. Selecting the **validate domain model** option causes the data service to parse the CQL query against the domain model, ensuring that all aspects of the query remain within the confines of the exposed domain model. Using these options together ensures that every query that reaches the CQL query processor implementation is both syntactically correct and conforms to the domain model, which can substantially simplify checking for potential errors in the query processor.

Auditing Tab

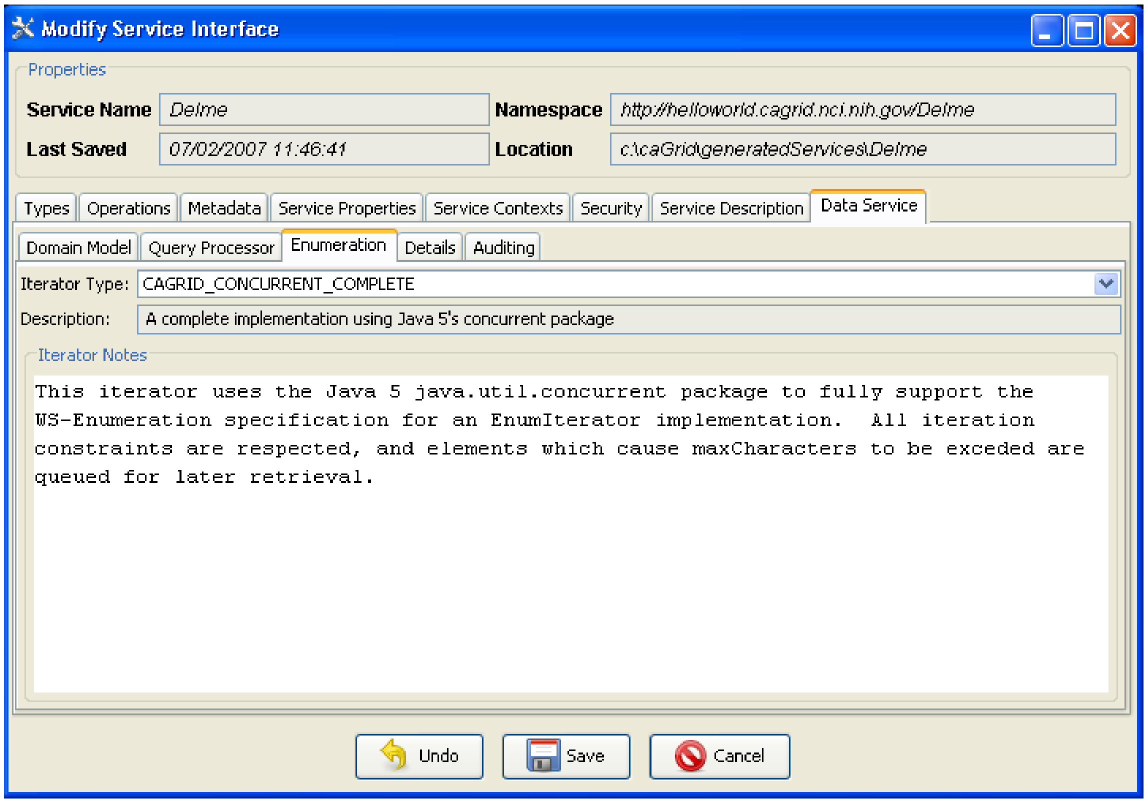
The Auditing tab allows auditing settings to be configured for the data service. Multiple auditors may be added to the service, each of which may handle auditing events in its own way. Multiple instances of the same auditor type may be added as well, allowing configuration options to dictate their behavior and handling of auditing events.

Four points of the data service query process may be audited:

- Query Begins
    - This event is fired when a query is first submitted to the data service before any validation or processing has been done.
- Validation Failure
    - This event is fired when a query fails the validation process. If validation is not enabled, this event will never be fired.
- Query Processing Failure
    - If a query fails to process correctly, this event is fired. The reason (exception) causing the failure is included in this event.
- Query Results
    - When a query completes successfully, this event is fired. The results of the query are available at this point as well.

Enumeration Tab

The Enumeration tab only displays when a data service's enumeration or BDT support features are enabled (Figure 4-3). This tab allows selecting a server side implementation to support the WS-Enumeration specification. In caGrid 1.1, there are five implementations which may be selected. Two are supplied with the Globus WS-Enumeration support, and three are part of caGrid itself. The default selection is the caGrid implementation which uses Java 5's concurrent package to fully support the specification. The other implementations support fewer features, but may be desirable for duplicating functionality of other WS-Enumeration enabled services.



Service Name: Delme
Namespace: http://helloworld.cagrid.nci.nih.gov/Delme
Last Saved: 07/02/2007 11:46:41
Location: c:\caGrid\generatedServices\Delme

Iterator Type: CAGRID_CONCURRENT_COMPLETE
Description: A complete implementation using Java 5's concurrent package

```
This iterator uses the Java 5 java.util.concurrent package to fully support the
WS-Enumeration specification for an EnumIterator implementation.  All iteration
constraints are respected, and elements which cause maxCharacters to be exceded are
queued for later retrieval.
```

Figure 4-3 Enumeration Implementation Selection

Functionality of the Extension

The Introduce Toolkit allows for extensions to be included in the service development process which add functionality to almost any step of the service build process. The Data Service extension makes use of three of these extension points. The first extension point used is immediately following service creation. This post-creation operation makes the following changes to the generated service:

- The data service WSDL file is copied into the service.
  - Optionally, if WS-Enumeration and/or BDT support is enabled, the WSDLs for these are included as well.
- The data service schemas for CQL and Domain Models are copied into the generated service.
- The data service libraries are copied in to the service.
- The base data service query method is added, and defined to be implemented in the copied libraries and provided in the copied WSDL.
  - If WS-Enumeration and/or BDT support is enabled, specialized query methods for each of these are added.
- Data Service specific service properties are added.

The next step in the build process added by the Data Service extension is invoked when modifications to the service are saved. This operation happens before the standard operations provided by Introduce are executed which generate code for user defined methods, edit WSDL files, and copy schemas. This pre-code generation operation makes the following changes to the service:

- Service properties are modified.
  - The property defining the query processor class implementation is populated with the user's defined value.
  - Properties required by the query processor implementation for initialization are created and added to the service.
- Domain model metadata is added.
  - The caDSR grid service is contacted to generate a domain model. This process can be time consuming depending on the network connection to the caDSR and the specific package and project combination selected.
  - If the service developer has defined an XML file containing the domain model definition, it is copied into the service.

Following the execution of the pre-code generation extension, the standard Introduce build operations are invoked. When this is complete, the final extension operation is invoked on the new service. This operation modifies the generated Eclipse files to add the new jar library files to the project's classpath.

Most Current Information

For the most current information regarding the Introduce extension for caGrid Data Services, please see the caGrid.org wiki page: *(http://www.cagrid.org/mwiki/index.php?title=Data_Services:Introduce_Extension:1.1)*.

Chapter 5 Developing Client Applications

This chapter introduces the client applications for caGrid services.

Topics in this chapter include:

Overview

Extending beyond the basic grid infrastructure, caBIG specializes grid technologies to better support the needs of the cancer research community. A primary distinction between basic grid infrastructure and the requirements identified in caBIG and implemented in caGrid is the attention given to data modeling and semantics. caBIG adopts a model-driven architecture best practice and requires that all data types used on the grid are formally described, curated, and semantically harmonized. These efforts result in the identification of common data elements, controlled vocabularies, and object-based abstractions for all cancer research domains. caGrid leverages existing NCI data modeling infrastructure to manage, curate, and employ these data models. Data types are defined in caCORE UML and converted into ISO/IEC 11179 Administered Components, which are in turn registered in the Cancer Data Standards Repository (caDSR). The definitions draw from vocabulary registered in the Enterprise Vocabulary Services (EVS), and their relationships are thus semantically described.

In caGrid, both the client and service APIs are object-oriented, and operate over well-defined and curated data types. Clients and services communicate through the grid using respectively Globus grid clients and service infrastructure. The grid communication protocol is XML, and thus the client and service APIs must transform the transferred objects to and from XML. This XML serialization of caGrid objects is restricted in that each object that travels on the grid must do so as XML, which adheres to an XML schema registered in the Global Model Exchange (GME). As the caDSR and EVS define the properties, relationships, and semantics of caBIG data types, the GME defines the syntax of the XML serialization of them. Furthermore, Globus services are defined by the Web Service Description Language (WSDL). The WSDL describes the various operations the service provides to the grid. The inputs and outputs of the operations, among other things, in WSDL are defined by XML schemas (XSDs). As caBIG requires that the inputs and outputs of service operations use only registered objects, these input and output data types are defined by the XSDs, which are registered in the GME. In this way, the XSDs are used both to describe the contract of the service and to validate the XML serialization of the objects that it uses. Figure 5-1 details the various services and artifacts related to the description of and process for the transfer of data objects between client and service.

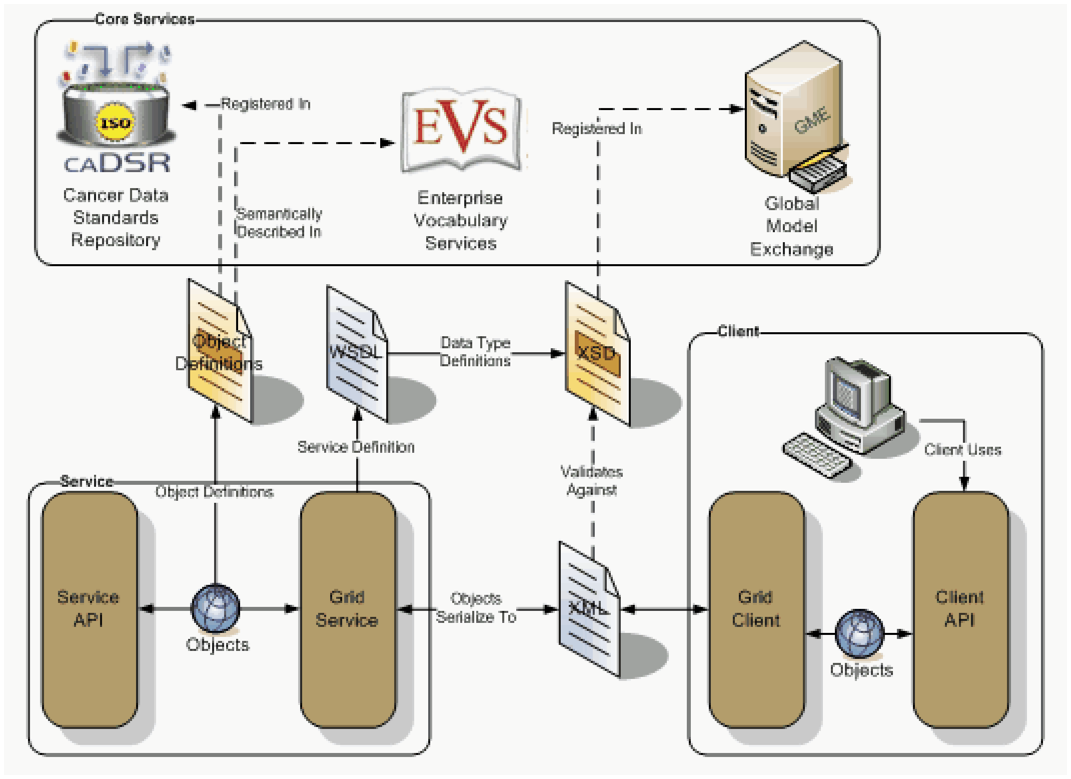


Figure 5-1 Data Description Overview

caGrid Client APIs

caGrid services are standard WSRF (Web Service Resource Framework) services typically implemented using the Globus toolkit version 4. While it is expected most clients and services will not only use Globus, but will also use the caGrid-provided tools that build on Globus, it is worth noting that caGrid uses an open specification for all communication between client and service. This enables interoperability between toolkits and programming languages, when needed. The extent of the caGrid user and programmer documentation focuses on the Java APIs provided by caGrid and, in some cases, Globus. Users interested in lower level specifications can consult the caGrid Specifications.

Secure Communication

Grid security can be complex and a detailed discussion on it is out of the scope of this document. The Globus documentation (http://www.globus.org/toolkit/docs/4.0/security/) and tutorials (http://gdp.globus.org/gt4-tutorial) provide a good overview and details on the topic. For the most part, caGrid clients and users need not concern themselves with all the details, but should have a basic understanding of what is happening "under the hood", and should understand how to "log in" and obtain credentials for secure communication with services.

caGrid builds on GSI (Globus Security Infrastructure), and uses Public Key Cryptography (PKI). Both services and clients may optionally have credentials (certificates), and authenticate and authorize each other. Services and corresponding clients generated from the Introduce toolkit attempt to automatically configure security appropriately, and this behavior is sufficient for most users, however it can be overridden (either by manually configuring this client "stub" or by overriding the provided *configureStubSecurity* method). Introduce clients attempt to communicate anonymously with services, as long as the service allows it (as advertised via its caGrid ServiceSecurityMetadata). If a service does not allow anonymous communication, client credentials must be used to authenticate the service. Introduce-created clients attempt to use the default Globus credentials, if present (via a grid-proxy-init, or logging in with Dorian and specifying setting the credentials as the defaults). Alternatively, Introduce-created clients have constructors, which take credentials (*GlobusCredential*), and also have an appropriate setter method (*setProxy*), which can be used after construction. Some services have different security requirements for different operations, so it may not be immediately obvious whether or not credentials are required. As such, when communicating with secure services it is good practice to have a valid grid proxy set as default, or specified on the client; the client APIs will only use it if necessary. For additional information on how to obtain grid credentials and access your grid proxy, see Chapter 6. For additional information on lower-level security details (such as how clients may perform authorization of services, or configure the communication channel), see GTS and the Globus Toolkit on page 83.

Definition of an EPR

As caGrid is a service-oriented architecture, the majority of the APIs made available are either tools and utilities, or client APIs for communicating with services. There are a number of caGrid-provided “core” services, as well as community provided Data and Analytical services. In order for a client to communicate with a service, it must first know its network end point, or address. In WSRF, this end point is referred to as an End Point Reference, or EPR. A detailed discussion of WSRF and EPRs is out of the scope of this document, but suffice it to say an EPR contains the information necessary to communicate with a service, and optionally identify a resource in that service. EPRs generally take two forms: a resource-qualified end point and a non-resource qualified end point (basically the URL of the service). In caGrid, all services can be communicated, at least initially, using the latter, which means clients that know the URL of the service may manually create an appropriate EPR instance. Complex services that manage state on behalf of the client (such as the workflow service and federated query service) have some operations that return a resource-qualified EPR, which can then be used to communicate with an appropriate service-side resource. However, again note, the initial communication with the service originates with a simple URL. Some caGrid APIs that communicate with services will provide convenience methods that take a string representation of a URL and, “under the hood”, construct an appropriate EPR. Other methods may require an EPR instance, but it can generally be constructed by just specifying the URL. For client applications, the source of the EPR is either created from a well-known URL (such as the address of the Index Service), or discovered at runtime using the Discovery API.

Obtaining an EPR for a Service

As mentioned above, the first step in communicating with a caGrid service is obtaining an appropriate EPR (though Introduce-generated clients do provide the shorthand constructors that simply take a string representation of the service’s URL). Often the address of a service of interest is not a “well known” value, and is something that is discovered at runtime. caGrid provides the means to discover services of interest by querying a live registry of available caGrid services. All caGrid services are required to publish standard metadata (described in the caGrid Metadata Design Document) that describes their functionality. This information is aggregated in the aforementioned registry (Index Service), and can be used to find out information about the currently running services, including their current EPRs. Building on this information, a Discovery API is provided with caGrid that facilitates the querying of this information toward the aim of discovering service EPRs. An overview of this process is shown in Figure 5-2.

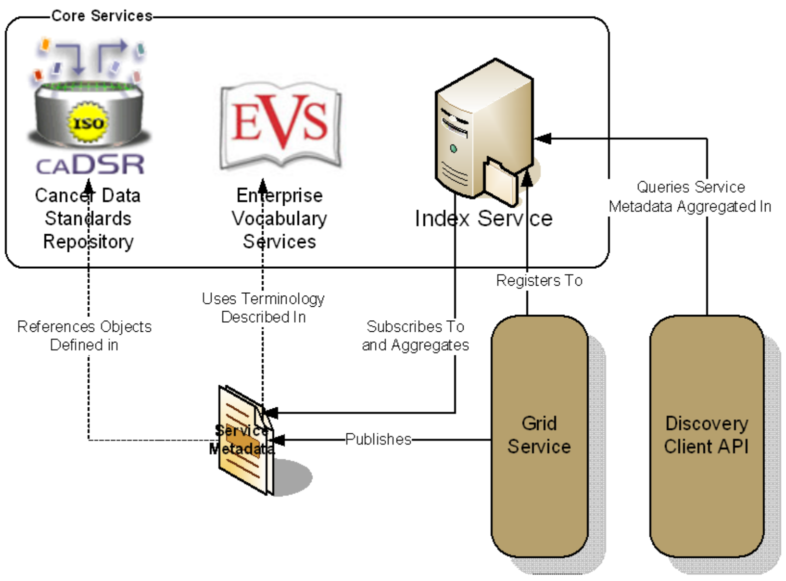
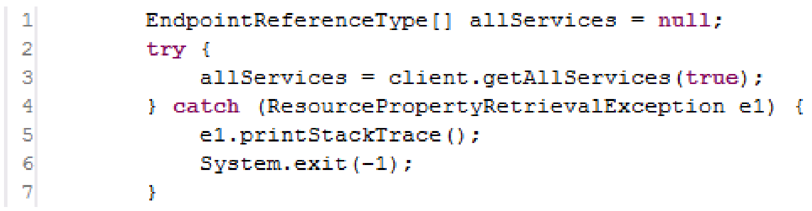

Core Services
ISO
caDSR
Cancer Data Standards Repository
EVS
Enterprise Vocabulary Services
Index Service
Queries Service Metadata Aggregated In
Registers To
References Objects Defined in
Uses Terminology Described In
Subscribes To and Aggregates
Service Metadata
Publishes
Grid Service
Discovery Client API

Figure 5-2 caGrid Advertisement and Discovery Overview

The Discovery API is intended to be used by any applications or services that wish to consume of data, analytics, and core services provided by caGrid. While there are still cases when interacting with a *particular instance* of a service is desired, the Discovery API provides a means by which applications can locate services by the information or capabilities they provide. One of the key advantages of the grid approach to caBIG is the dynamic discovery of available resources.

In order to make use of the Discovery API, the discovery process must be "bootstrapped" using a well-known service address of an Index Service. The default constructor of the *DiscoveryClient*, the main interface to the Discovery API, should default to the official NCI Index Service. However, this behavior can be modified by using the constructor that takes the Index Service URL, or by calling the appropriate setter method (*setIndexEPR*). Additional details on this, as well as all Discovery API information, can be found in the Discovery section of the caGrid 1.1 Programmer's Guide.

The simplest discovery scenario, shown in Figure 5-3, is to just query the Index Service for all registered services. The boolean value specified in line 3, indicates whether services should be ignored if they do not expose the caGrid standard metadata. In most application scenarios, a value of "true" is used, as services without standard metadata are either: not compliant, not properly configurable, or inaccessible (e.g. behind a misconfigured firewall).



```
1        EndpointReferenceType[] allServices = null;
2        try {
3            allServices = client.getAllServices(true);
4        } catch (ResourcePropertyRetrievalException e1) {
5            e1.printStackTrace();
6            System.exit(-1);
7        }
```

Figure 5-3 Discovering All Services

As shown in the example on line 3, the method returns an array of EPRs. This is true of all discovery operations. The EPRs in the array represent the services matching the specified criteria (in this case just that it is a valid caGrid service), and can be used to create clients to invoke operations on the corresponding services (detailed later).

There are many discovery operations available in the DiscoveryClient. They provide a range of capabilities from “full text search” suitable for a freeform webpage-like interface, simple text-based criteria such as specifying operation names or concept code, and complex criteria (“query by example”) such as specification of point of contact information or UML class criteria.

While there are many discovery methods that take a UMLClass prototype, to discover services based on data types, an example is shown below in Figure 5-4. This method, *discoverServiceByOperationInput*, locates services that provide an operation that takes, as input, an instance of the specified data type. The example below shows services that provide operations that take caBIO’s Gene instances as input. This prototype object can be as partially populated as desired (such as only specifying the package name, or being more explicit in specifying the exact project name and version).

```
        EndpointReferenceType[] services = null;
        try {
            UMLClass umlClass = new UMLClass();
            umlClass.setClassName("Gene");
            umlClass.setPackageName("gov.nih.nci.cabio.domain");

            services = client.discoverServicesByOperationInput(umlClass);
        } catch (ResourcePropertyRetrievalException e) {
            e.printStackTrace();
            System.exit(-1);
        }
```

Figure 5-4 Discover Services by Input

Additionally, there are methods to discover services by “type”. For example, there are several methods named like *discoverDataServices**, which only return services that implement the standard Data Service operations. Services may also be discovered by identifying the concept code matching the service type of interest, and invoking the *discoverServicesByConceptCode* method, which searches for services based on concepts applied to the service. There is a concept representing “Grid Service” in the ontology and derived concepts such as “Analytical Grid Service” and “Data Grid Service”. It is expected additional concepts will be derived in the future, as driven by the community.

Inspecting a Service’s Metadata

Depending on how specific the discovery criteria which was used to discover services, it is possible there will be many services returned, and it may be necessary to find out additional information about the matching services in order to select which one should be used.

The Metadata API provides the ability to obtain a Java bean representation of standard metadata by simply providing an EPR of the service of interest (such as those returned from the discovery methods). The main interface to this API is the *MetadataUtils* class, which contains a number of static methods. An example of using this API, shown below in Figure 5-5, demonstrates accessing a service’s standard *ServiceMetadata*, which is common to all caGrid services. As described above, the first step is to obtain an appropriate EPR (line 1). Given this EPR, the *MetadataUtils*’s *getServiceMetadata* method, shown on line 4 in Figure 5-5, can be used to obtain the bean representation of the metadata. Upon successful completion of this method, the fully populated bean can be inspected to obtain the information of interest. Several exceptions, sub classed from the base *ResourcePropertyRetrievalException*, can be thrown by this operation. A non-discriminating client may choose simply to handle this base exception. Additional details on the other exceptions, and why they may be throw, is described in the Metadata section of the caGrid 1.1 Programmer's Guide.

```
EndpointReferenceType serviceEPR = /* Some service's EPR */
try {
    ServiceMetadata serviceMetadata =
        MetadataUtils.getServiceMetadata(serviceEPR);
} catch (InvalidResourcePropertyException e) {
    // TODO handle the case where the service doesn't expose
    // standard metadata
    e.printStackTrace();
} catch (RemoteResourcePropertyRetrievalException e) {
    // TODO handle the case where there was some other problem
    // communicating with the service (unavailable, untrusted, etc)
    e.printStackTrace();
} catch (ResourcePropertyRetrievalException e) {
    // TODO handle all other general error (such as inability to
    // deserialize the result)
    e.printStackTrace();
}
```

Figure 5-5 Accessing Standard Service Metadata

Given an instance of *ServiceMetadata*, all information required by caGrid standard metadata can be inspected. This and all of its fields are standard, logic-less Java beans, and can be inspected by invoking the appropriate *getters*. Additionally, the Metadata API provides the capability to write the instances to XML for storage or display.

Details of the content of the metadata can be found in the caGrid Metadata design document, as well as an overview in the Metadata section of the caGrid 1.1 Programmer's Guide, but it is worth noting, as shown in Figure 5-2, a majority of the standard metadata is derived from extracting information from the caDSR and EVS. As such, the caDSR grid service, and the EVS grid service can also be used, respectively, to find out additional information about the data types and semantics relevant to the service. For example, many aspects described in the metadata (services, operations, classes, attributes, etc) have associated *SemanticMetadata* items, which describe the semantics of the item, including its EVS-maintained concept code. This code can be used to locate the concept in EVS and navigate the ontology, determining further semantic relevance. As another example, the metadata about each operation's input and output define the caDSR registered Project from which they came. The caDSR grid service can be used to find out additional information about that Project, and the rest of its model.

Invoking Operations on a Service

The end goal of discovering services and inspecting their metadata is generally to select an appropriate service and invoke operations it provides. This may be the execution of analytical routines, querying for data, or invocation of a core caGrid service.

While the grid makes it possible to dynamically invoke services for which a client has no APIs (and this is true for caGrid services), this is generally not the procedure clients or applications follow (clients interested in this, however, may search the web for “dynamic service invocation”). Generally, the API for the service is already available, and just “bound” to the particular service of interest at runtime. For example, to query any caGrid Data Service, a common client API can be used, regardless of the type of data it exposes. Applications built to query data services generally would build against this API. For Analytical Services, however, it is more likely a client API specific to a type of analytical service would be used, and, again, instances of that service would be bound to it at runtime. In both cases, the application developer would just make use of a pre-provided client API. The caGrid infrastructure makes this process as simple as using a using a local API; each provided client APIs takes a service’s EPR or address in its constructor, and then the API’s methods can simply be invoked. The APIs take care of all of the grid communication, handling security, and XML serialization and deserialization. All caGrid core service APIs are provided with caGrid, and when a service is built using Introduce, a client API for that service is also created. It is expected a common location for service clients will be available for caBIG (on GForge). All the examples that show the communication with services provided in this document or the caGrid 1.1 Programmer’s Guide are examples of such APIs. In the absence of these client APIs, more limited “stub” client APIs can also be generated by the grid tools by downloading the service’s WSDL (the Globus documentation provides some details on this).

One instance where a client or application may wish to invoke operations on a service without having previously downloaded a client API is the construction of a workflow. The caGrid workflow infrastructure provides the mechanism to describe service invocations using a workflow language (BPEL), and request the workflow service perform the invocation. Further details on the workflow infrastructure can be found in the caGrid 1.1 Programmer’s Guide.

Client Application Case Study: caArray

caArray is an open source microarray data management system that allows users to submit, annotate and download microarray data. caArray was developed using the caBIG compatibility guidelines, as well as the Microarray Gene Expression Data (MGED) society standards for microarray data. Compatibility with these standards and guidelines will facilitate data sharing and integration of diverse data types including clinical, imaging, tissue and functional genomics data. A number of analytical tools that connect to caArray are already available, including geWorkbench and GenePattern that both provide a variety of data analysis, visualization and annotation functions for microarray and other data types. The caArray MAGE-OM Grid Service API is a caGrid Data Service that exposes the functionality of MAGE-OM API over the grid. The Grid Service provides access to data in the caArray database via a web service call issued to a dedicated Grid Service server at NCI or any other site with an accessible caArray MAGE-OM Grid Service installation.

caArray Discovery Example

Discovery is performed by querying the Index Service for a list of Endpoint References (Figure 5-6). The Discovery Client is identified by the Index Service Endpoint Reference constructed from a URL; here the caTRIP Index Service is pointed to. It provides a number of utility methods for querying the Index Service. The example shows getting all of the services and then querying by the Domain Model of caArray (Figure 5-7)

```
String url = "http://cagrid01.bmi.ohio-state.edu:8080/wsrf/services/DefaultIndexService";
DiscoveryClient dclient = new DiscoveryClient(url);
for (EndpointReferenceType epr : dclient.getAllServices(false)) {
  System.out.println(epr);
}

EndpointReferenceType[] eprs = dclient.discoverDataServicesByDomainModel("caArray");
System.out.println(eprs[0]);
```

Figure 5-6 caArray Discovery Example

```
Address: http://140.254.80.174:8080/wsrf/services/cagrid/GridImageResultService
Address: https://152.16.96.128:8443/wsrf_catissue_newintr/services/cagrid/CaTissueCore_Full
Address: http://140.254.80.50:50015/wsrf/services/cagrid/DICOMDataService

. . .

Address: http://caarraydb-stage.nci.nih.gov:80/wsrf/services/caGrid/CaArraySvc
```

Figure 5-7 Results from the caArray Discovery Example

caArray Metadata Example

Service-level metadata can be retrieved from any caGrid 1.1 grid service. This includes information such as the publisher of the service, the methods exposed, and the domain model used. The service is identified by an Endpoint Reference and metadata is retrieved using the MetadataUtils class (Figure 5-8). The service-level metadata can be retrieved, as well as the domain model. This example demonstrates getting the display name of the hosting research center, as well as the domain model project name (Figure 5-9).

```
String url = "http://caarraydb-stage.nci.nih.gov:80/wsrf/services/caGrid/CaArraySvc";
EndpointReferenceType epr = new EndpointReferenceType(new URI(url));

ServiceMetadata metadata = MetadataUtils.getServiceMetadata(epr);
System.out.println(metadata.getHostingResearchCenter().getResearchCenter().getDisplayName());

DomainModel model = MetadataUtils.getDomainModel(epr);
System.out.println(model.getProjectShortName());
```

Figure 5-8 caArray Metadata Example

```
NCI Center for Bioinformatics
<ns1:DomainModel projectDescription="This model makes up a the portion of caArray . . .
caArray
```

Figure 5-9 Results from the caArray Metadata Example

caARRAY Invocation Example

A service client can be constructed by passing in a URL to the endpoint of the service or an EPR (Figure 5-10). The client exposes all of the methods exposed by the service. Input parameters should be constructed and passed into the desired method, and the output of the method is returned (Figure 5-11). This example demonstrates querying the caArray service for a specific microarray experiment (Figure 5-12).

```
public CaArraySvcClient(String url) throws MalformedURIException, RemoteException

public CaArraySvcClient(EndpointReferenceType epr) throws MalformedURIException, RemoteException
```

Figure 5-10 Two constructors for the caTRIP Tumor Registry data service client

```
CQLQuery query = new CQLQuery();
Object target = new Object();
target.setName("gov.nih.nci.mageom.domain.Experiment.Experiment");
query.setTarget(target);
Attribute att = new Attribute();
att.setName("identifier");
att.setPredicate(Predicate.EQUAL_TO);
att.setValue("gov.nih.nci.ncicb.caarray:Experiment:1015897558050098:1");
target.setAttribute(att);

String url = "http://caarraydb-stage.nci.nih.gov:80/wsrf/services/caGrid/CaArraySvc";
CaArraySvcClient client = new CaArraySvcClient(url);
CQLQueryResults results = client.query(query);
results.setTargetClassname(
  "gov.nih.nci.cagrid.caarray.stubs.mageom.domain.experiment.Experiment"
);
CQLQueryResultsIterator iter = new CQLQueryResultsIterator(results);
while (iter.hasNext()) {
  Experiment experiment = (Experiment) iter.next();
  System.out.println(experiment.getName());
}
```

Figure 5-11 caArray Invocation Example

```
Gene Expression in Ovarian Cancer Reflects Both Morphology and Biological Behavior
```

Figure 5-12 Results from the caArray Invocation Example

Chapter 6 caGrid Security

This chapter describes the caGrid security infrastructure, which provides services and tools to administer and enforce security policy.

Topics in this chapter include:

Overview

The Grid Authentication and Authorization with Reliably Distributed Services (GAARDS) infrastructure serves as the caGrid 1.1 Security Infrastructure (Figure 6-1). GAARDS provides services and tools for the administration and enforcement of security policy in an enterprise Grid. GAARDS was developed on top of the Globus Toolkit and extends the Grid Security Infrastructure (GSI) to provide enterprise services and administrative tools for: 1) grid user management, 2) identity federation, 3) trust management, 4) group/VO management, 5) access control policy management and enforcement, and 6) integration between existing security domains and the grid security domain.

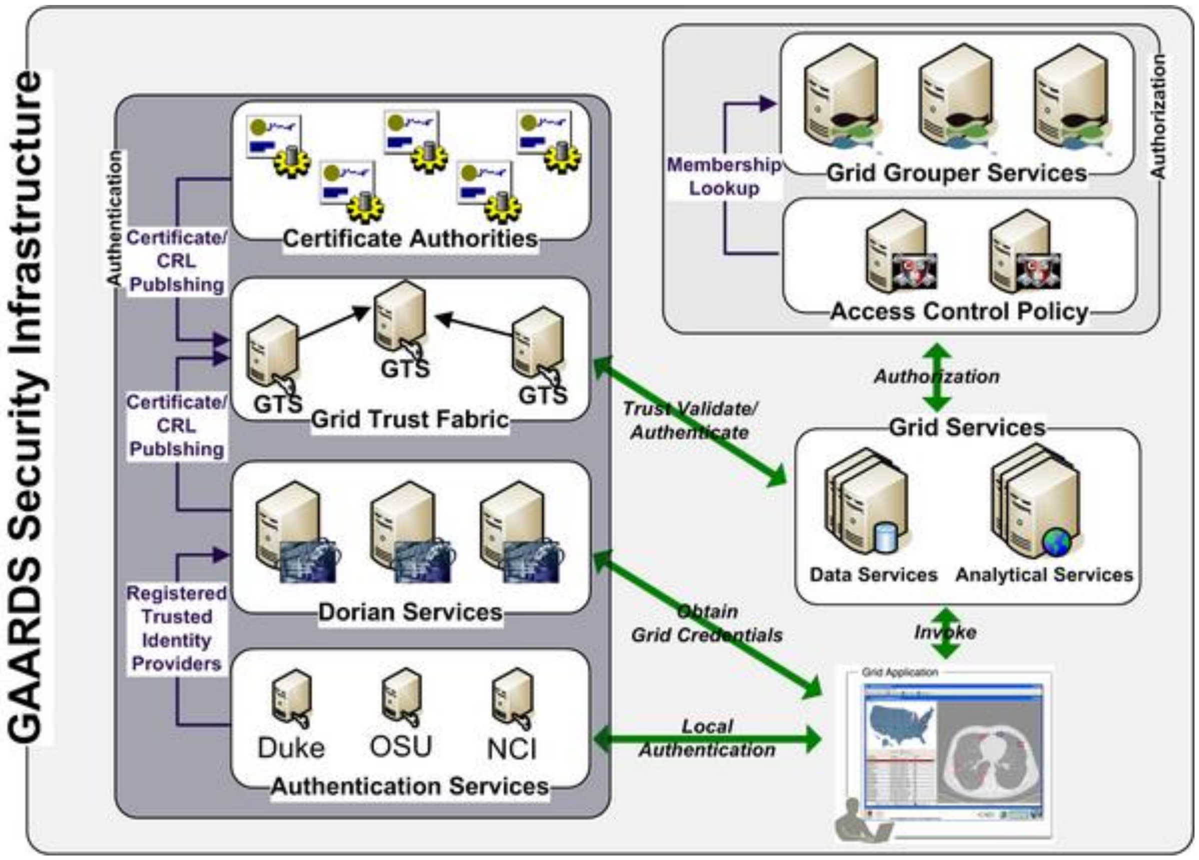


Figure 6-1 GAARDS Security Infrastructure

Figure 6-1 illustrates the GAARDS security infrastructure. In order for users and applications to communicate with secure services, grid credentials are required, which include a Grid User Account. Dorian is a grid user management service that 1) hides the complexities of creating and managing grid credentials from users and 2) provides a mechanism for users to authenticate using their institution's authentication mechanism, assuming a trust agreement is in place between Dorian and the institution. Dorian provides two methods for registering for a grid user account: 1) register directly with Dorian 2) have an existing user account in another security domain. It is anticipated that most users will use their existing locally provided credentials for obtaining grid credentials and only users that are unaffiliated with an existing credential provider should register directly with Dorian. In order to use an existing user account to obtain grid credentials, the existing credential provider must be registered in Dorian as a Trusted Identity Provider. It is anticipated that the majority of grid user accounts will be provisioned based on existing accounts. The advantages to this approach are: 1) users can use their existing credentials to access the grid and 2) administrators only need to manage a single account for a given user. To obtain grid credentials, Dorian requires proof or a SAML assertion (see the Dorian Design Guide for more details) that proves that the user is locally authenticated. The GAARDS Authentication service provides a framework for issuing SAML assertions for existing credential providers such that they may be used to obtain grid credentials from Dorian. The authentication service also provides a uniform authentication interface in which applications can be built. Figure 6-1 illustrates the process for obtaining grid credentials.

1. The user/application first authenticates with their local credential provider via the

authentication service and obtains a SAML assertion as proof they authenticated.

2. They then use the SAML assertion provided by the authentication service to obtain grid credentials from Dorian. Assuming the local credential provider is registered with Dorian as a trusted identity provider and that the user's account is in good standing, Dorian issues grid credentials to the user. It should be noted that the use of the authentication service is not required; an alternative mechanism for obtaining the SAML assertion required by Dorian can be used. If as user is registered directly with Dorian and not through an existing credential provider, they may contact Dorian directly for obtaining grid credentials.

Once a user has obtained grid credentials from Dorian, they may invoke secure services. Upon receiving grid credentials from a user, a secure service authenticates the user to ensure that the user has presented valid grid credentials. Part of the grid authentication process is verifying that grid credentials presented were issued by a trusted grid credential provider (for example, Dorian or other certificate authorities). The Grid Trust Service (GTS) maintains a federated trust fabric of all the trusted digital signers in the grid. Credential providers such as Dorian and grid certificate authorities are registered as trusted digital signers and regularly publish new information to the GTS. Grid services authenticate grid credentials against the trusted digital signers in a GTS (shown in Figure 6-1).

Once the user has been authenticated, a secure grid service determines if a user is authorized to perform what they requested. Grid services have many different options available to them for performing authorization. The GAARDS infrastructure provides two approaches that can each be used independently or can be used together. It is important to note any other authorization approach can be used in conjunction with the GAARDS authentication/trust infrastructure. The Grid Grouper service provides a group-based authorization solution for the grid, wherein grid services and applications enforce authorization policy based on membership to groups defined and managed at the grid level. Grid services can use Grid Grouper directly to enforce their internal access control policies. Assuming the authorization policy is based on membership to groups provisioned by Grid Grouper; services can determine whether a caller is authorized by simply asking Grid Grouper whether the caller is in a given group.

The Common Security Module (CSM) is a more centralized approach to authorization. CSM is a tool for managing and enforcing access control policy centrally. Access control policies can be based on membership to groups in Grid Grouper. Grid services that use CSM for authorization simply ask CSM if a user can perform a given action. Based on the access control policy maintained in CSM, CSM decides whether or not a user is authorized. In Figure 6-1, the grid services defer the authorization to CSM. CSM enforces its group based access control policy by asking Grid Grouper whether the caller is a member of the groups specified in the policy.

GAARDS Administration User Interface

The GAARDS Administration User Interface (Admin UI) is for administrating Dorian, Grid Grouper, and the Grid Trust Service (GTS). The Common Security Module (CSM) is administered through a separate interface called the CSM User Provisioning Tool (UPT).

Launch the GAARDS Admin UI by typing `ant security` from the installation directory. The

GAARDS Admin UI is pre-configured to run with a default list of settings bound to the distribution being used. The default configuration can be modified by editing the UI configuration file (`[`*`Installation Directory`*`]/projects/security-ui/etc/security-ui-conf`). The remainder of this chapter describes how to administer each of the GAARDS services.

Grid User and Host Management

Managing users and provisioning accounts in the grid is complex. The Globus Toolkit implements support for security via its Grid Security Infrastructure (GSI). GSI utilizes X.509 Identity Certificates for identifying a user. An X.509 Certificate with its corresponding private key forms a unique credential or so-called "grid credential" within the grid. Since grid credentials are long term credentials and are not directly used in authenticating users to the grid, a short term credential called a *grid proxy* is used. *Grid proxies* consist of a private key and corresponding long term certificate signed by the long term grid credential private key. A *Grid proxy* is an extension to traditional X.509 certificates providing the ability to delegate your credentials to other services, as in the case of workflow. Although this approach is effective and secure, it is difficult to manage in a multi-institutional environment. Using the base Globus toolkit, the provisioning of grid credentials is a manual process, which is far too complicated for users. The overall process is further complicated if a user wishes to authenticate from multiple locations, as a copy of their private key and certificate has to be present at every location. Not only is this process complicated, securely distributing private keys is error prone and poses a security risk. There are also many complexities in terms of provisioning user accounts in an environment consisting of tens of thousands of users from hundreds of institutions, each of which most likely has a user account at their home institution.

Dorian is a grid service for the provisioning and management of grid users accounts. Dorian provides an integration point between external security domains and the grid, allowing accounts managed in external domains to be federated and managed in the grid. Figure 6-2 illustrates an example usage scenario for Dorian. To obtain grid credentials or a proxy certificate, users authenticate with their institution using the institution's conventional mechanism. Upon successfully authenticating the user, the local institution issues a digitally signed Secure Access Markup Language (SAML) assertion, vouching that the user has authenticated. The user then sends this SAML assertion to Dorian in exchange for grid credentials. Dorian will only issue grid credentials to users that supply a SAML assertion from a *Trusted Registration Authority*. For example, in Figure 6-2, when a Georgetown user wishes to invoke a grid service that requires grid credentials, he first supplies the application with his username and password to the Georgetown credential provider as he would normally do. The application client authenticates the Georgetown user with the Georgetown credential provider, receives a signed SAML assertion which it subsequently passes to Dorian in exchange for grid credentials. These credentials can then be used to invoke the grid services. This illustrates how Dorian can leverage an institution's existing authentication mechanism and bring its users to the grid.

To facilitate smaller groups or institutions without an existing credential provider, Dorian also has its own internal credential provider (which is registered as a Trusted Registration Authority). This allows users to authenticate to Dorian directly, thereby enabling them to access the grid. The Dorian credential provider provides administrators with facilities for approving and managing users. All of the Dorian functionality is made available through a grid service interface. Figure 6-2 illustrates a scenario of a client using the Dorian IdP to authenticate to the Grid. In this scenario, the unaffiliated user wishes to invoke a grid service. Given that this unaffiliated user has registered and been approved for an account, she is able to authenticate with the Dorian IdP by supplying her username and password. Upon successfully authenticating the user, the Dorian IdP issues a SAML Assertion just like institutional IdPs, which can be presented to Dorian in exchange for grid credentials. The credentials can be used to invoke the grid service.

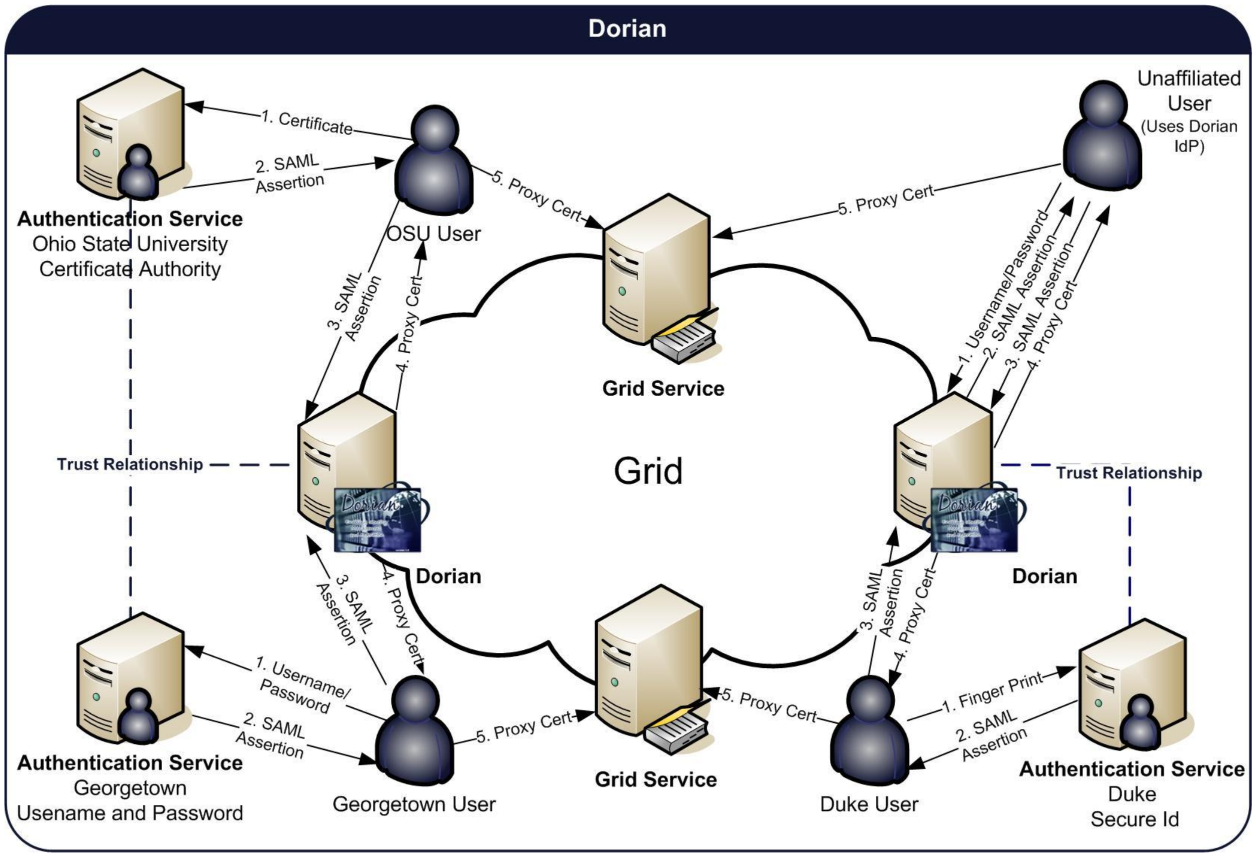


Figure 6-2 Dorian

Registration Authorities

The institutional credential providers that Dorian is configured to trust are referred to as Trusted Registration Authorities (TRA). Dorian only creates credentials for users whose identity assertions are signed by an TRA. The set of TRAs can be managed by Dorian administrators through its grid service interface. The Dorian grid service interface provides functionality for adding, modifying, and removing TRAs, which consist of the following: Id, Name, Status, User Policy, Certificate, and acceptable authentication methods. The Id is a unique id assigned by Dorian to identify the registration authority. The Name is assigned by an administrator and provides a human readable name to easily identify a registration authority. The Status specifies the current status of the registration authority: Active or Suspended. Users associated with a "suspended" registration authority will be refused access to Dorian and will be listed in the CRL of the Dorian CA. Each registration authority is associated with a set of configurable User Policies that are applied to each user when they authenticate. These policies designate how Dorian should handle users from a specified registration authority. As an example, a policy might dictate what to do when a new user tries to create grid credentials for the first time. An automatic approval policy would automatically register the user with Dorian and create a grid account for the user. A manual approval policy would automatically register the user but not enable the grid account until an administrator manually approves it. User policies can also be used to dictate what to do when a user's grid credentials expire. For example, an automatic renewal policy would enable automatic creation of a new set of credentials using the Dorian certificate authority, whereas a manual renewal policy would require an administrator to do so. The User Policy framework is extensible; administrators can implement local policies.

Each registration authority must also specify its own certificate. When Dorian receives a SAML assertion signed by a registration authority it verifies that the assertion was signed with the private key that corresponds to the registration authority's certificate. Finally, each registration authority must be configured with a list of acceptable authentication methods. A SAML authentication assertion specifies the method in which the credential provider authenticated the user. In order for the SAML assertion to be accepted by Dorian, the authentication method specified in the assertion must be specified as acceptable in the corresponding registration authority.

Account and Certificate Creation

When a user first attempts to create a grid proxy using Dorian, a grid user account is created for them. The account includes user information, user status, user role, and a set of grid credentials including the associated grid identity. The user information includes the user's local institution id, the id of the registration authority the user is associated with, and an email address. The user's status corresponds to the user's current status: Active, Suspended, Pending, or Expired. Only users with an "Active" status may access Dorian. A user's role specifies whether or not the grid user is a Dorian administrator. Only administrators may access the administrative functionality to manage trusted registration authorities or to manage grid accounts. A user's grid credentials consist of a certificate and private key, signed by the Dorian CA that are used by Dorian to issue grid proxy certificates. A user's grid identity is comprised of the Certificate Authority's Subject DN (Distinguished Name), the registration authority Id, and the user's id at his institution. When a user's grid account is created the initial status of the account is "Pending". As mentioned earlier, if the registration authority has an Auto Approval User Policy in place, the status will automatically be changed to "Active", giving the user instant access to Dorian. Administrators can update a user's status and role, and can renew a user's credentials.

Grid Proxy Certificate Creation

Users authenticate with grid services using grid proxy certificates. Such a grid "proxy" is a short-term credential (private key and certificate) that is created from a user's long-term grid credentials. Dorian facilitates the creation of grid proxies for its users. To create a grid proxy the user supplies a proxy lifetime and the SAML assertion provided by their credential provider to the Dorian client. The Dorian client generates a new public/private key pair and sends the proxy lifetime, public key, and SAML assertion to the Dorian Grid Service. The Dorian Grid Service validates the SAML assertion and employs the user's previously stored grid credentials (long term certificate and private key) to create and sign a proxy certificate for the user-supplied public key. The proxy certificate is then returned to the user. The proxy certificate and locally generated private key can then be used as a grid proxy credential to invoke secure grid services. It is important to note that throughout this process no sensitive information, that is, private keys, are passed over the network.

Host Certificate Creation

In order to run secure services securely, the container hosting the services must run with a host credential. A host credential consists of an X.509 certificate and a private key. The Dorian issues host certificates to users who possess a user certificate issued by the Dorian CA. Valid users may request a host certificate from Dorian. To request a host certificate a user must 1) authenticate with Dorian using their grid proxy 2) specify a host name for the certificate, and 3) generate an RSA public/private key pair which will make up the host credentials. From the key pair the public key is sent to Dorian as part of the request and the private key should be securely maintained by the user. All certificate requests require approval of a Dorian administrator. If a Dorian administrator approves the certificate request, a host certificate will be created and signed with the Dorian CA private key. The host certificate contains the public key provided by the user and together with the private key securely maintained by the user will make up a host credential. Each host certificate issued by Dorian is bound to a user or owner, generally the users that requested it; however an administrator may assign a new owner. If the owner's account is revoked, compromised, or suspended, any host certificates bound to them will be suspended as well.

Installation and Configuration

To install and configure Dorian, use the following steps.

Step 1: Installation Software Prerequisites

Table 6-1 lists the software prerequisites for Dorian.

Software	Version	Description
Java SDK	Jsdk1.5 or higher	The GTS is written in Java and requires the Java SDK. After installing, set up an environmental variable pointing to the Java SDK directory and name it JAVA_HOME.
MySQL	Mysql 4.1.x or higher	For persisting the trust fabric and other information.
Ant	Ant 1.6.5	The GTS service along with the Globus Toolkit, on which the GTS is built, uses Jakarta Ant for building and deploying.
Globus	Globus 4.0.3	The GTS is built on top of the Globus Toolkit. The

Software	Version	Description
		GTS requires the ws-core installation of the Globus Toolkit.
Tomcat (Only required if deploying to Tomcat)	Tomcat 5.0.28	The GTS can be optionally deployed as a Grid Service to a Tomcat deployed Globus Toolkit.

Table 6-1 Dorian Software Prerequisites

Step 2: Building Dorian

Dorian is distributed as a standalone project as well as part of other projects such as caGrid. Each of the distributions contains a *dorian* directory herein referred to as DORIAN_LOCATION. To build Dorian type `ant clean all` from the DORIAN_LOCATION directory.

Note: Depending on the Dorian distribution it may be required to build the entire project that Dorian is distributed with prior to building Dorian. For example if you have obtained a caGrid distribution this is required; if you received a Dorian standalone distribution this is not required.

Step 3: Configuring Dorian

Dorian is configured through a single configuration file which is located at DORIAN_LOCATION/etc/dorian-conf.xml. For simple deployments only the following configuration elements need to be modified.

1) Database Configuration
2) CA Subject Name

Database Configuration

Dorian uses a MySQL database to persist account information. Dorian must be modified such that it will interact with your MySQL database. To modify the database configuration to interact with your database, set the values of the *host*, *port*, *username*, and *password* elements.

Certificate Authority Subject Name

Dorian manages an internal certificate authority for signing user and host certificates. The certificate authority is created the first time the service is started. It is important that the subject of the CA certificate is unique and meaningful to your deployment. To set the subject of the certificate authority for you deployment edit the *CASubject* element. The default value of *C=US,O=abc,OU=xyz,OU=caGrid,CN=caGrid Dorian CA* is provided as an example.

Note: The configuration changes specified thus far are the minimum configurations required for simple deployments of Dorian. Complete details on configuring Dorian can be accessed by consulting the following website:

http://www.cagrid.org/mwiki/index.php?title=Dorian:1.1:Administrators_Guide:Configuration

These details on this site include configuring some of the more advanced features such as using a Hardware Security Module (HSM) for the storage of keys or for details on integrating Dorian with the Grid Trust Service (GTS).

Step 4: Obtaining Host Credentials for Dorian

Dorian requires that it runs as a secure service. In order to run a secure service, the container hosting the service must run with a host credential. A host credential consists of an X.509 certificate and a private key. One of the features Dorian provides is the ability to issue and manage host credentials. Although you may host a credential elsewhere, Dorian has a command line utility that can be used to issue a host credential for the container that it will run in. To leverage this command line utility type the following from a command prompt:

```
%> cd DORIAN_LOCATION
%> ant createDorianHostCredentials
```

You will immediately be prompted for the name of the host that will be running Dorian. Enter the host name and press ENTER. At the prompt, enter a directory where the host certificate and private key should be written and press ENTER. The utility creates a host certificate and private key for Dorian and inform you where on the file system they were written. The entire output of the program is shown below:

```
$ ant createDorianHostCredentials
Buildfile: build.xml

setGlobus:

checkGlobus:
     [echo] Globus: C:\ext\ws-core-4.0.3

createDorianHostCredentials:
    [input] Please enter the host:
somehost.example.com
    [input] Please enter the directory to write out the host
credentials:
c:/certificates
     [java] /C=US/O=abc/OU=xyz/OU=caGrid/OU=Dorian IdP/CN=dorian
     [java] Successfully created the host certificate:
     [java] Subject:
C=US,O=abc,OU=xyz,OU=caGrid,OU=Services,CN=host/somehost.ex
ample.com
     [java] Created: Thu Jun 21 19:21:45 EDT 2007
     [java] Expires: Sat Jun 21 19:21:45 EDT 2008
     [java] Succesfully wrote private key to
c:\certificates\somehost.example.co
m-key.pem
     [java] Succesfully wrote certificate to
c:\certificates\somehost.example.co
m-cert.pem

BUILD SUCCESSFUL
Total time: 29 seconds
```

Once the host credentials are obtained, configure Globus to trust the Dorian Certificate Authority that issued those credentials by typing the following from the command prompt:

```
%> cd DORIAN_LOCATION
%> ant configureGlobusToTrustDorian
```

Upon completion a similar output to the following should display:

```
$ ant configureGlobusToTrustDorian
Buildfile: build.xml

setGlobus:

checkGlobus:
     [echo] Globus: C:\ext\ws-core-4.0.3

configureGlobusToTrustDorian:
     [java] Succesfully configured Globus to trust the Dorian CA:
C=US,O=abc,OU=
xyz,OU=caGrid,CN=caGrid Dorian CA
     [java] Succesfully wrote CA certificate to
C:\Users\jdoe\.globus\certif
icates\2d45eee5.0
     [java] Succesfully wrote CA signing policy to
C:\Users\jdoe\.globus\cer
tificates\2d45eee5.signing_policy

BUILD SUCCESSFUL
Total time: 5 seconds
```

Step 5: Configuring a Secure Container

Once host credentials are obtained, use them to configure a secure container. Dorian can be run from a secure Globus container or a secure Tomcat container. For directions on how to configure a secure Globus container, please consult the following website:

http://www.cagrid.org/mwiki/index.php?title=CaGrid:How-To:SecureGlobusContainer

For directions on how to configure a secure Tomcat container please consult the following website:

http://www.cagrid.org/mwiki/index.php?title=CaGrid:ConfigureTomcat

Step 6: Deploying Dorian

Once a secure container is configured (Globus or Tomcat), deploy Dorian to that container. To deploy Dorian to a secure Globus container, type the following from a command prompt:

```
%> cd DORIAN_LOCATION
%> ant deployGlobus
```

To deploy Dorian to a secure Tomcat container, type the following from a command prompt:

```
%> cd DORIAN_LOCATION
%> ant deployTomcat
```

Regardless of which container is selected, a significant amount of output should be printed to the screen. If the deployment is successful, *"BUILD SUCCESSFUL"* is output to the screen.

Step 7: Verifying the Installation

Once Dorian is deployed, the installation and configuration of Dorian is complete. Before verifying that the installation was successful, start the Dorian service by starting the container that Dorian was deployed to. For directions on starting a secure Globus container, please consult the following website:

http://www.cagrid.org/mwiki/index.php?title=CaGrid:How-To:SecureGlobusContainer

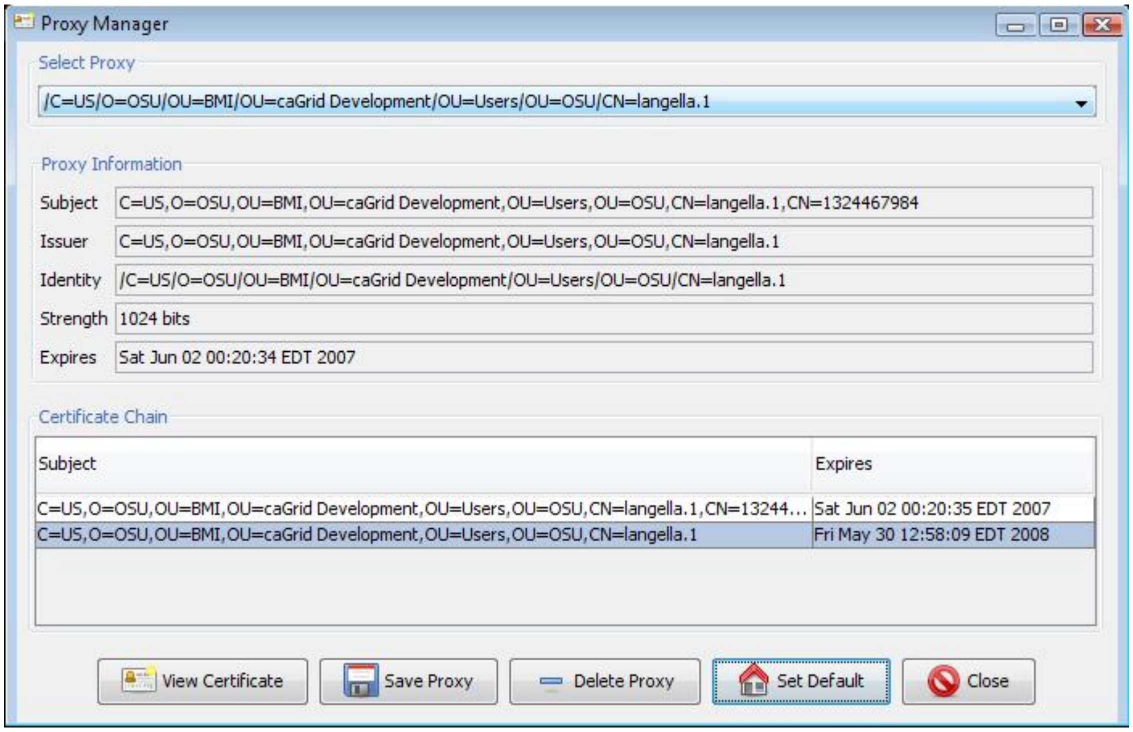
To start a secure Tomcat container run the startup script (`startup.sh` or `startup.bat`) located in TOMCAT_INSTALLATION_DIRECTORY/bin. When the container starts, verify that the Dorian installation was successful by typing the following from the command prompt:

```
%> cd DORIAN_LOCATION
%> ant ui
```

Complete the following steps in the Dorian Administration UI that opens:

1. Click **Login**. In the Login screen, select the following:
   a. From the **Dorian Service** drop down menu: https://localhost:8443/wsrf/services/cagrid/Dorian.
   b. From the **Authentication Service** drop down menu: https://localhost:8443/wsrf/services/cagrid/Dorian.
   c. In the **User Id** text box enter *dorian.*
   d. In the **Password** text box enter *password.*
2. Click **Authenticate**.

You will be logged onto Dorian using the default administrator (dorian). If the login is successful another window, similar to the one in Figure 6-3, should open containing the details of your credential. This indicates Dorian has been successfully installed and configured.



Select Proxy: /C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=OSU/CN=langella.1

Subject	C=US,O=OSU,OU=BMI,OU=caGrid Development,OU=Users,OU=OSU,CN=langella.1,CN=1324467984
Issuer	C=US,O=OSU,OU=BMI,OU=caGrid Development,OU=Users,OU=OSU,CN=langella.1
Identity	/C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=OSU/CN=langella.1
Strength	1024 bits
Expires	Sat Jun 02 00:20:34 EDT 2007

Subject	Expires
C=US,O=OSU,OU=BMI,OU=caGrid Development,OU=Users,OU=OSU,CN=langella.1,CN=13244...	Sat Jun 02 00:20:35 EDT 2007
C=US,O=OSU,OU=BMI,OU=caGrid Development,OU=Users,OU=OSU,CN=langella.1	Fri May 30 12:58:09 EDT 2008

Figure 6-3 Dorian credentials window

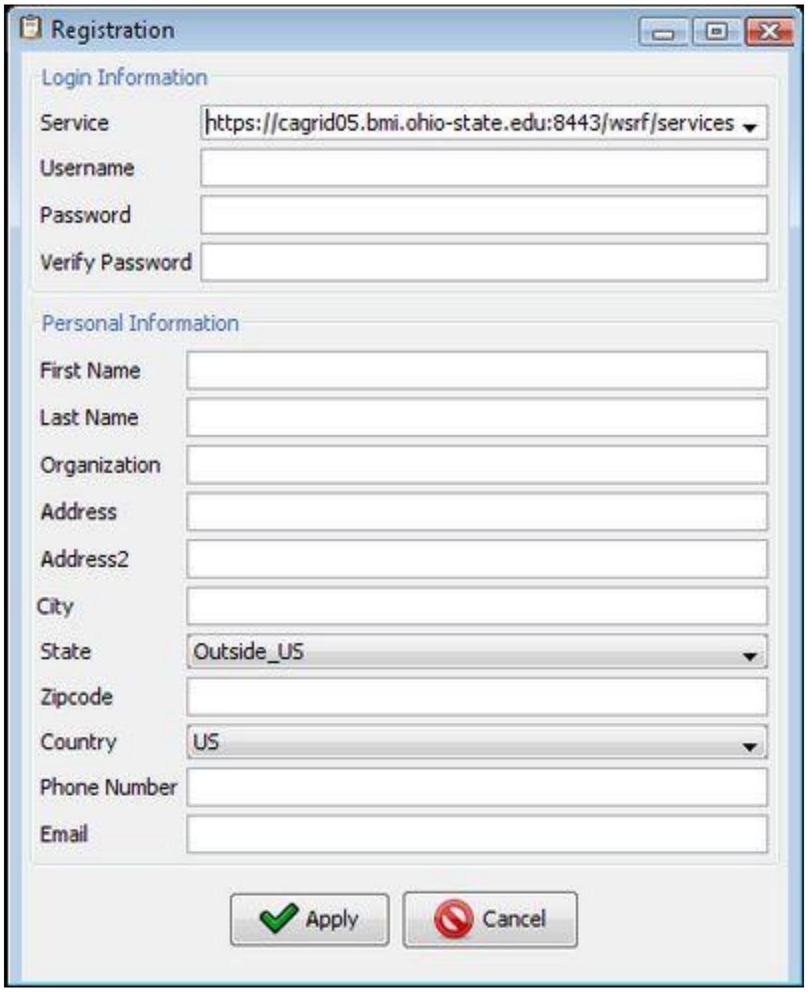
Registering for an Account with the Dorian IdP

It is anticipated that most users will use their existing locally provided credentials for obtaining grid credentials and only users that are unaffiliated with an existing credential provider should register directly with Dorian. The Dorian Identity Provider (DorianIdP) gives developers, smaller groups, research labs, unaffiliated users, and other groups that do not have their own IdP, the ability to leverage Dorian. The DorianIdP provides a method for prospective users to register for an account. When users register they create a user id and password which they can subsequently use to authenticate with the Dorian IdP. When a user authenticates, the Dorian IdP provides the user with a SAML assertion, which can then be used to authenticate with Dorian's to create grid proxies. The DorianIdP provides mechanisms for administrators to manage users; this includes modifying user information (name, address, email, etc.), changing passwords, granting and revoking access, and other administrative actions. All operations provided by the Dorian IdP are made available through Dorian's grid service interface. Administrative operations require administrators to authenticate with a trusted grid proxy. The GAARDS UI provides a method for perspective users to register with the Dorian IdP. To register with the DorianIdP through the GAARDS UI, use the following steps:

1. From the main menu in the GAARDS UI, select **User Management > Local Account > Registration**.
2. To register, select the URI of the Dorian you wish to register with. Next specify a username and password; this will be the username and password that you use to

authenticate with the Dorian IdP (Figure 6-4).

3. Finally enter your personal information and click **Apply**. In most cases your account will need to be approved by an administrator before you will be able to login. Depending on the policies of your administrator, you may be contacted once your account has been approved as the Dorian IdP does not provide an automated method of contacting you.



Service: https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services
State: Outside_US
Country: US

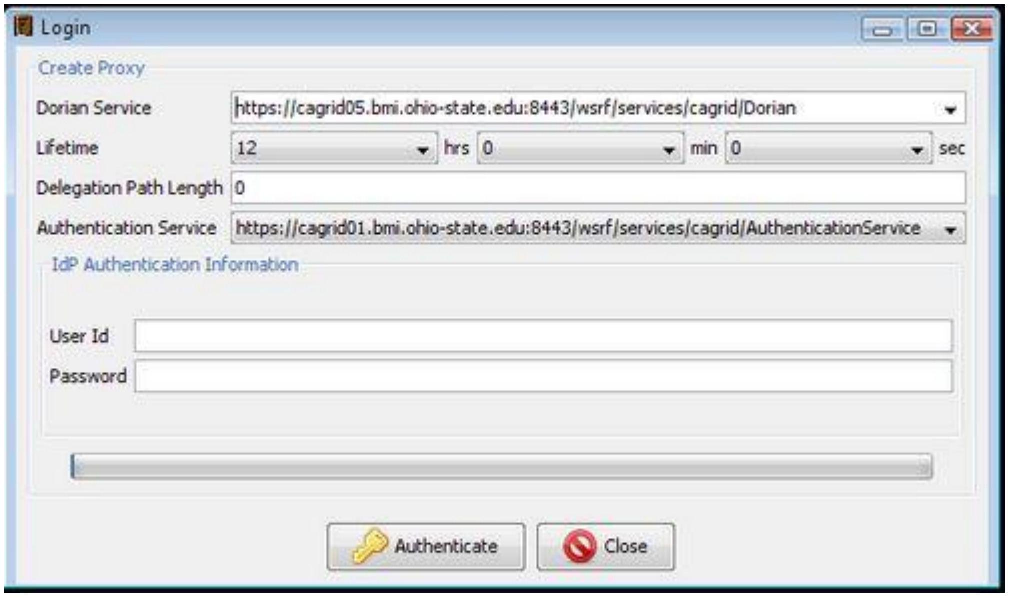
Figure 6-4 Registration window

Logging onto the Grid

The GAARDS UI provides the ability to create and manage grid proxies and credentials. To obtain grid credentials, use the following steps.

1. Click the **Login** button on the toolbar in the GAARDS UI to open the Create Proxy or login window (Figure 6-5).
2. To login, specify the Dorian that maintains your grid user account by selecting the Dorian URI from the **Identity Federation Service** drop down menu.
3. The GAARDS UI is pre-configured with a list of Dorians through its configuration file. If the Dorian you wish to select is not in the list, enter it.

4. Select the lifetime of your grid proxy; this is how long your credentials are good for. Select from the **Lifetime** drop down menus for hours, minutes, and seconds.
5. Specify how many times your credentials can be delegated. Delegating your credentials gives another party the ability to act on your behalf or as you. For example, if you allow a delegation path length of 1, you allow a grid service you connect with to connect to another grid service as you. However, the second grid service would not be able to connect to another grid service as you. By default the delegation path length is set to 0, and in most cases it will not need to be increased. To increase the delegation path length, change the **Delegation Path Length** text field.
6. Specify the Authentication Service you wish to authenticate with by selecting the URI from the **Authentication Service** drop down menu. After selecting the Authentication Service you will be supplied with input fields to enter information you need to authenticate. Provide the information requested. In Figure 6-5 a Dorian IdP is selected. Since the Dorian IdP requires a user id and password, input fields display. The GAARDS UI is pre-configured with a list of Identity Providers. If your Identity Provider is not listed in the **Identity Provider** drop down menu, add it by editing the GAARDS UI configuration file.
7. Once you have entered the required IdP Authentication Information, click **Authenticate** to 1) authenticate you with your Identity Provider, 2) obtain a SAML Assertion from your Identity Provider, and 3) contact Dorian using the SAML Assertion to facilitate the creation of a grid proxy. Once the grid proxy is created the Create Proxy window closes and the Proxy Manager window opens with the newly created proxy shown. The Proxy Manager window allows the management of grid proxies or grid credentials that you locally created. For details on the Proxy Manager window, refer to the next section Managing Grid Credentials.
8. Click **Set Default**.



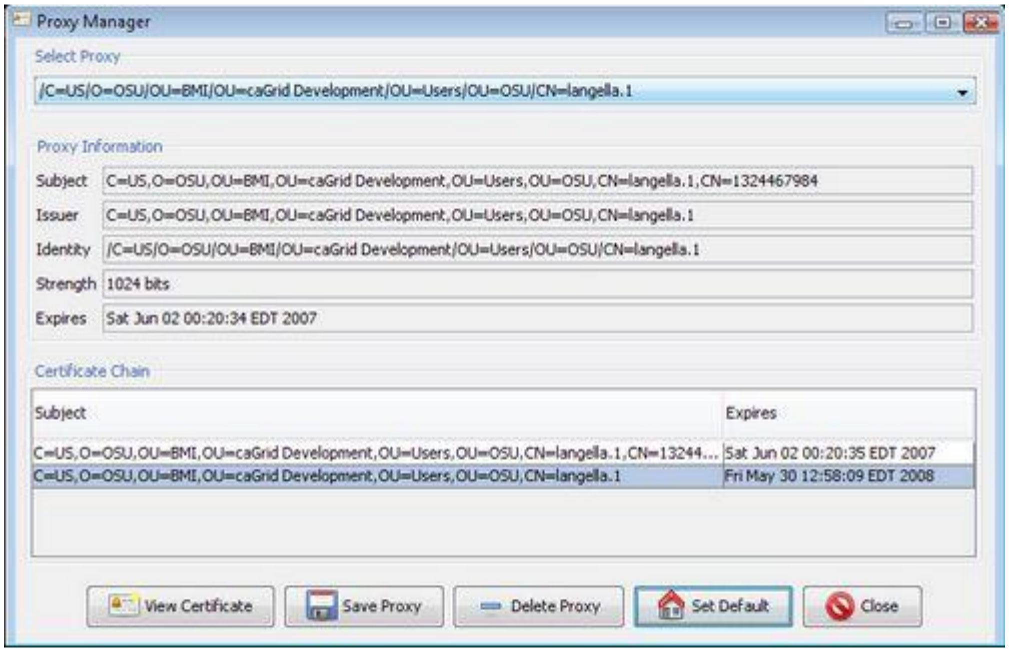
Dorian Service: https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian
Lifetime: 12 hrs 0 min 0 sec
Delegation Path Length: 0
Authentication Service: https://cagrid01.bmi.ohio-state.edu:8443/wsrf/services/cagrid/AuthenticationService
User Id
Password

Figure 6-5 GAARDS UI login window

Managing Grid Credentials

The Proxy Manager window allows the management of grid proxies or grid credentials that have been locally created. This window is accessible after logging into Dorian or it is directly accessible through the GAARDS UI as follows.

1. Click the **Credential Manager** button on the toolbar in the GAARDS UI.
2. The **Select Proxy** drop down menu contains a list of all the non-expired proxies that you created with the addition of the default proxy (Figure 6-6). Generally Globus clients use the default proxy to connect to grid services if no other proxy is specified. To set the default proxy, select the proxy you wish to make the default from the **Select Proxy** drop down menu and click **Set Default**. Selecting a proxy from the drop down menu displays some information about the proxy as well as the certificate chain for the proxy. The Proxy information includes the subject of the proxy certificate, the issuer of the proxy certificate, the grid identity, the strength of the proxy certificate, and when the proxy expires. The certificate chain table lists each certificate in the proxies certificate chain, with the proxy certificate listed first. View the details of a certificate in the chain by selecting it and by clicking **View Certificate**.
3. Finally, delete a proxy by selecting it from the **Select Proxy** drop down menu and click **Delete Proxy**.



/C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=OSU/CN=langella.1

Subject	C=US,O=OSU,OU=BMI,OU=caGrid Development,OU=Users,OU=OSU,CN=langella.1,CN=1324467984
Issuer	C=US,O=OSU,OU=BMI,OU=caGrid Development,OU=Users,OU=OSU,CN=langella.1
Identity	/C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=OSU/CN=langella.1
Strength	1024 bits
Expires	Sat Jun 02 00:20:34 EDT 2007

Subject	Expires
C=US,O=OSU,OU=BMI,OU=caGrid Development,OU=Users,OU=OSU,CN=langella.1,CN=13244...	Sat Jun 02 00:20:35 EDT 2007
C=US,O=OSU,OU=BMI,OU=caGrid Development,OU=Users,OU=OSU,CN=langella.1	Fri May 30 12:58:09 EDT 2008

Figure 6-6 Proxy Manager window

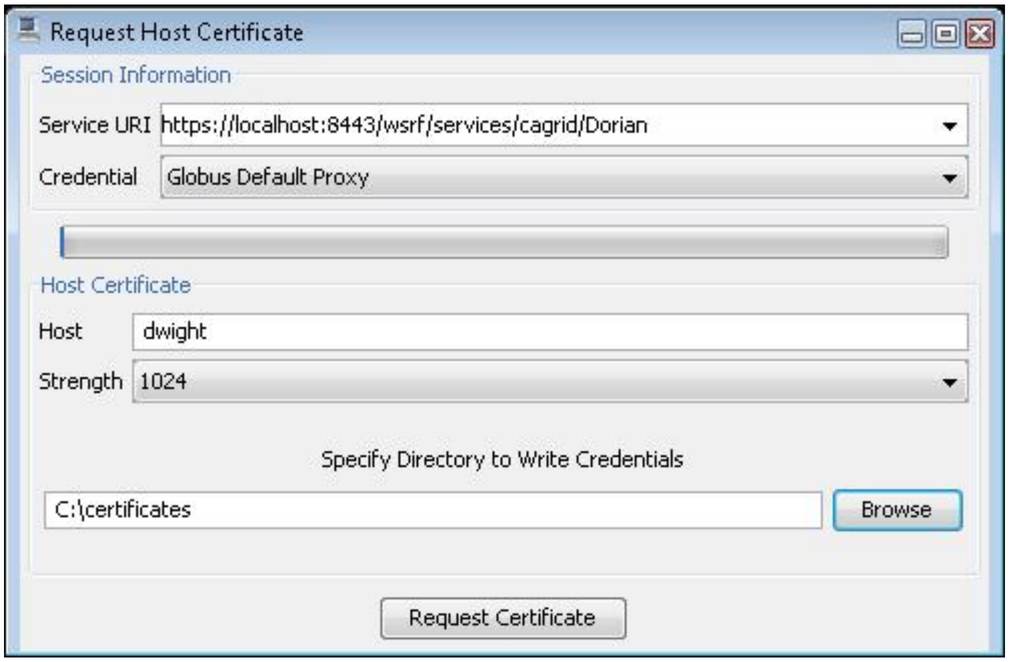
Requesting and Managing Host Credentials

In order to run secure services securely, the container hosting the services must run with a host credential. A host credential consist of an X.509 certificate and private key. Dorian provides a means for users with a grid user account to request a host credential for their services. To request a host credential, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account.
3. Select **MyAccount > Request a Host Certificate** to launch the Request a Host Certificate window (Figure 6-7).
4. From the **Service URI** drop down menu, select the URI of the Dorian you wish to request a host certificate from.
5. In the **Host** text box, enter the name of the host you are requesting host credentials for.
6. Specify the directory on the file system where the host credentials should be written by clicking **Browse**.
7. Click **Request Certificate**.

The UI submits the host certificate request to Dorian and ,upon receiving it, Dorian will either immediately approve the request or submit the request to the administrator for approval. When the request is immediately approved, the host credentials (certificate and private key) are written to the directory specified. The file containing the certificate is named *THE_HOSTNAME_YOU_ENTERED-cert.pem* and the file containing the private key is named

THE_HOSTNAME_YOU_ENTERED-key.pem.



Service URI: https://localhost:8443/wsrf/services/cagrid/Dorian
Credential: Globus Default Proxy
Host: dwight
Strength: 1024
C:\certificates

Figure 6-7 Request Host Certificate window

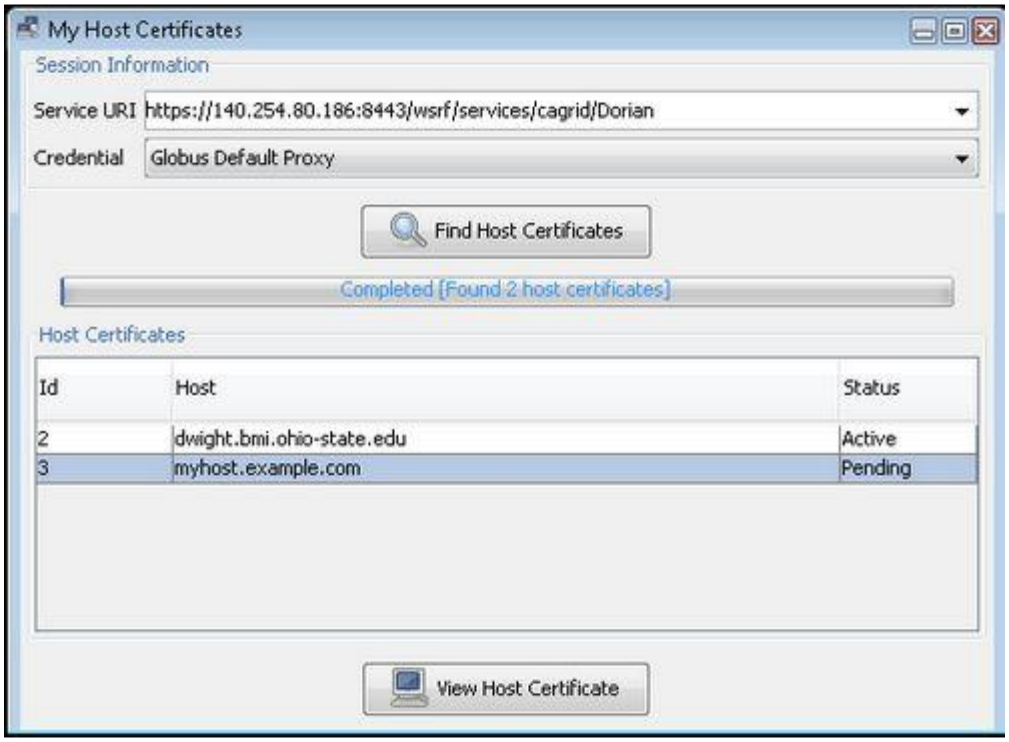
When a host certificate request requires approval of an administrator, the file containing the private key is *THE_HOSTNAME_YOU_ENTERED-key.pem*. The host certificate will not be written since it is not issued until the request is approved. The GAARDS Admin UI provides a means of checking the status of your host certificates request(s). To check the status of a host certificate request use the following steps:

1. Launch the GAARDS UI.
2. If you have not done so, logon to the Grid using your user account.
3. Select **MyAccount > My Host Certificates** to launch the My Host Certificates window.
4. From the **Service URI** drop down menu, select the URI of the Dorian you wish to request a host certificate from.
5. Click **Find Host Certificates**.

All the host certificates that are bound to your user account are listed. For each host certificate listed, the current status (Active, Suspended, Rejected, Pending, Suspended) of that host certificate displays. If the status of your certificate request is *Pending* your request has not yet been reviewed by an administrator. If the status is *Rejected* your request was rejected and a host certificate will not be issued. If the status is *Active* your request was approved by an administrator and your certificate is ready to be downloaded. To download and save your certificate to the file system, use the following steps:

1. Select the certificate from the table at the bottom of the My Host Certificates window (Figure 6-8).
2. Click **View Host Certificate** to open a window containing the details for the selected host certificate.

3. Click the **Certificate** tab.
4. Click **Save Certificate** to open a file browser window.
5. Browse to the directory you initially specified when you made the certificate request
6. In the **File name** text field enter *THE_HOSTNAME_YOU_ENTERED-cert.pem*
7. Click **Save**.



Service URI: https://140.254.80.186:8443/wsrf/services/cagrid/Dorian

Credential: Globus Default Proxy

Completed [Found 2 host certificates]

Id	Host	Status
2	dwight.bmi.ohio-state.edu	Active
3	myhost.example.com	Pending

Figure 6-8 My Host Certificates window

Administrating Dorian

Default Administrator Account

When Dorian is first run, it creates an administrative account with the username *dorian* within the Dorian IdP. By default the Dorian user is able to administer both the Dorian IdP and all aspects of the identity management and federation components of Dorian. The default password for the dorian account is *password*; it is recommended that you change the password immediately through the local user management interface of the Dorian IdP. In the future, you may wish to provision administrative rights to other or “real” users once they have Dorian accounts, at which time you can disable or revoke privileges on the dorian account.

Managing Trusted Identity Providers

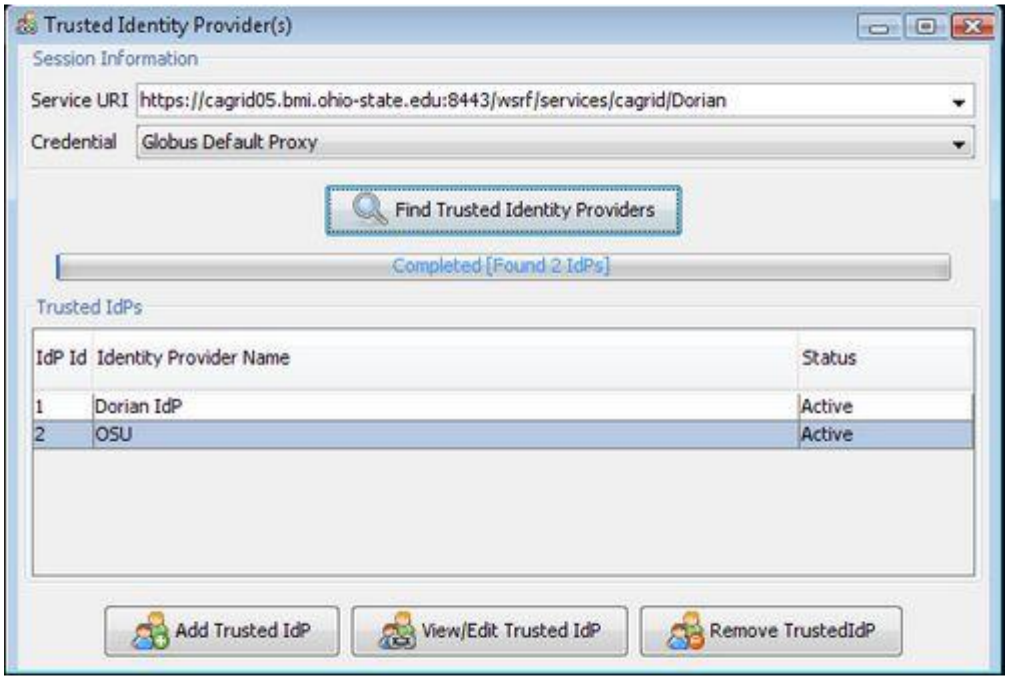
In order for Dorian to issue grid proxies to a user using their institution provided credentials, the institution's Identity Provider (IdP) must be registered with and trusted by Dorian. IdPs registered with and trusted by Dorian are referred to as Trusted Identity Providers (Trusted IdPs). The set of Trusted IdPs can be managed by Dorian administrators through the GAARDS UI, which provides the ability for remotely adding, modifying, and removing Trusted IdPs. A Trusted IdP consists of the following information: IdP Id, IdP Name, IdP Status, User Policy, Certificate, acceptable authentication methods, and attribute specifications. The IdP Id is a unique id assigned by Dorian to identify the IdP. The IdP name is assigned by an administrator and provides a human readable name to easily identify an IdP. The IdP Status specifies the current status of the IdP: Active or Suspended. The status of an IdP allows an administrator to easily grant or suspend access to the grid for all users associated with an IdP. Each Trusted IdP is associated with a set of configurable User Policies that are applied to each user when they authenticate. These policies designate how Dorian should handle users from a specified Trusted IdP. Policies generally dictate what to do when a new user is encountered and what to do when a user's long term certificate expires. Currently Dorian supports four policies:

1. **Auto Approval / Auto Renewal** – A new user is automatically registered and given access to the grid (user's status is active). When a user whose long term certificate expires, it is automatically renewed.
2. **Auto Approval / Manual Renewal** – A new user is automatically registered and given access to the grid (user's status is active). When a user whose long term certificate expires, an administrator is required to manually renew it.
3. **Manual Approval / Auto Renewal** – A new user is automatically registered but not granted access, and an administrator is required to grant access (user's status is pending). When a user whose long term certificate expires, it is automatically renewed.
4. **Manual Approval / Manual Renewal** – A new user is automatically registered but not granted access, and an administrator is required to grant access (user's status is pending). When a user whose long term certificate expires, an administrator is required to manually renew it.

When Dorian receives a SAML assertion from a Trusted IdP it verifies that the assertion was signed with the private key that corresponds to the Trusted IdP's certificate. Thus the Trusted IdP's certificate must be specified. Each Trusted IdP must be configured with a list of acceptable authentication methods. A SAML authentication assertion specifies the method in which the Trusted IdP authenticated the user. In order for the SAML assertion to be accepted by Dorian, the authentication method specified in the assertion must be specified as acceptable in the corresponding Trusted IdP. Dorian requires the SAML assertions provided by Identity Provider's to specify four attributes which are maintained by Dorian for each user, such that Dorian and its administrators may effectively administrate grid user accounts. These attributes include (1) user's local unique user id within the IdP, (2) user's first name, (3) user's last name, (4) user's email address. In a SAML Assertion, attributes are specified with a namespace and name. Because the naming of attributes may differ from IdP to IdP, Dorian does not place requirements on how the attributes are named within the SAML Assertion so long as the values of the attributes meet Dorian's formatting requirements. Therefore the namespace and name of each of the four attributes must be specified for each Trusted IdP, such that Dorian knows what to look for when it receives a SAML assertion from the IdP. To manage Trusted IdPs through the GAARDS UI, use the following steps.

1. Select **User Management > Grid Account Management > Trusted Identity Provider(s)** to open Trusted Identity Provider Management window (Figure 6-9). All the IdPs trusted by a Dorian are listed
2. From the **Service** drop down menu, select the service URI of the Dorian you wish to list the Trusted IdPs of. If it is not in the list enter it manually.
3. From the **Proxy** drop down menu, select the proxy or credentials to use to authenticate to Dorian. This must be a proxy of a Dorian administrator.
4. Click **Find Trusted Identity Providers**.

The Trusted IdPs are listed in the table below the progress bar. The list includes the Trusted IdP's id, human readable name, and status. In the example below, there are two Trusted IdPs listed; the first is Dorian's Local IdP and the second is the Ohio State University IdP. Thus in the example in Figure 6-9, Dorian would accept credentials from its local IdP and from the Ohio State University.


Trusted Identity Provider(s)

Session Information

Service URI https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian

Credential Globus Default Proxy

Completed [Found 2 IdPs]

Trusted IdPs

IdP Id	Identity Provider Name	Status
1	Dorian IdP	Active
2	OSU	Active

Figure 6-9 Trusted Identity Provider(s) window

Adding a Trusted Identity Provider

To add a Trusted IdP to Dorian, use the following steps.

1. In the Trusted Identity Provider Management window, click **Add Trusted IdP** to open the Add Trusted IdP window (Figure 6-10). The window has three tabs, each of which

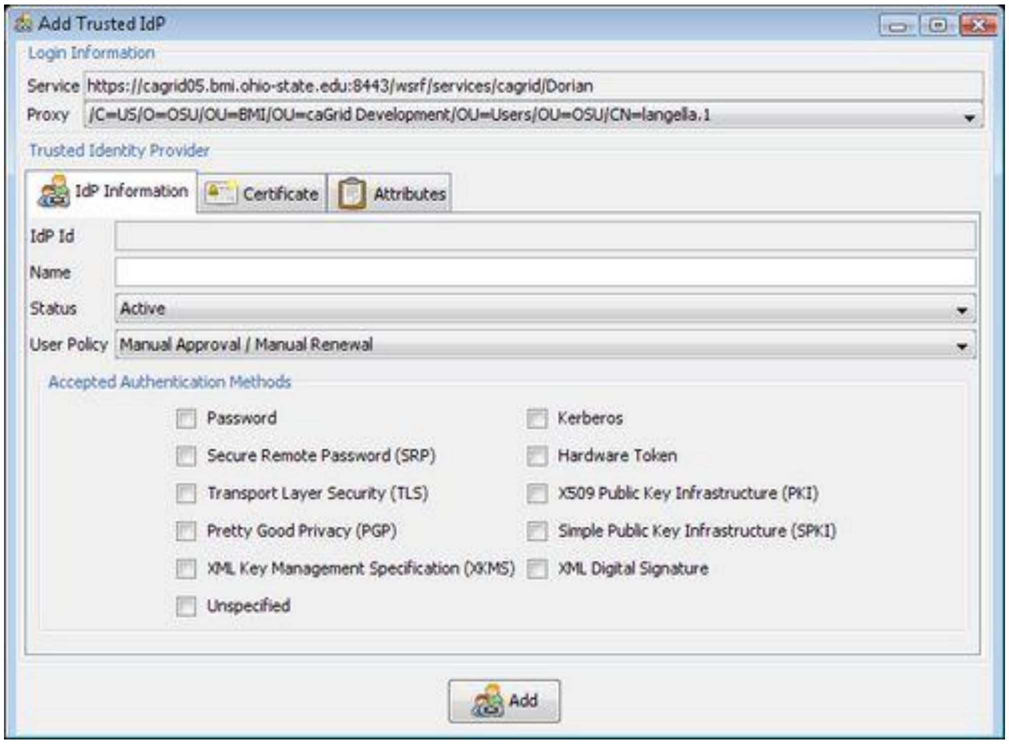
requires information to be specified.

   a. **IdP Information Tab -** specify the name, status, user policy, and acceptable authentication methods.

   b. **Certificate Tab -** specify the certificate that corresponds to the private key that is used by the IdP in signing SAML Assertions that is issues. The certificate must be specified in PEM format. Click the **Import Certificate** to open a file browser in to find the certificate.

   c. **Attributes Tab -** specify the namespace and name that the IdP uses for representing each of the four required attributes in its SAML assertions, such that Dorian knows how to retrieve the attributes from the IdP's SAML assertions.

2. Once you have specified all the required information, click **Add** to add the IdP to Dorian as a Trusted IdP. Assuming you set the status of the newly added IdP to active, Dorian immediately begins accepting SAML assertions from the IdP.


Add Trusted IdP

Login Information

Service: https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian

Proxy: /C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=OSU/CN=langella.1

Trusted Identity Provider

IdP Information | Certificate | Attributes

IdP Id:

Name:

Status: Active

User Policy: Manual Approval / Manual Renewal

Accepted Authentication Methods

- [ ] Password
- [ ] Kerberos
- [ ] Secure Remote Password (SRP)
- [ ] Hardware Token
- [ ] Transport Layer Security (TLS)
- [ ] X509 Public Key Infrastructure (PKI)
- [ ] Pretty Good Privacy (PGP)
- [ ] Simple Public Key Infrastructure (SPKI)
- [ ] XML Key Management Specification (XKMS)
- [ ] XML Digital Signature
- [ ] Unspecified

Figure 6-10 Add Trusted IdP window

Viewing and Updating a Trusted Identity Provider

To view and update a Trusted IdP, use the following steps.

1. In the Trusted Identity Provider Managemen*t* window, select the Trusted IdP of interest and click **View/Edit Trusted IdP** to open the Trusted IdP window. This window has three tabs:

   a. **IdP Information Tab -** updates the name, status, user policy, and acceptable authentication methods.

b. **Certificate Tab -** updates the certificate that corresponds to the private key that is used by the IdP in signing SAML Assertions that is issues. If you update the certificate it must be specified in PEM format. Click **Import Certificate** to open a file browser to find the certificate.

c. **Attributes Tab -** update the namespace and name that the IdP uses for representing each of the four required attributes in its SAML assertions.

2. Once you have finished updating the Trusted IdP's information, click **Update** to commit the changes, which take effect immediately.

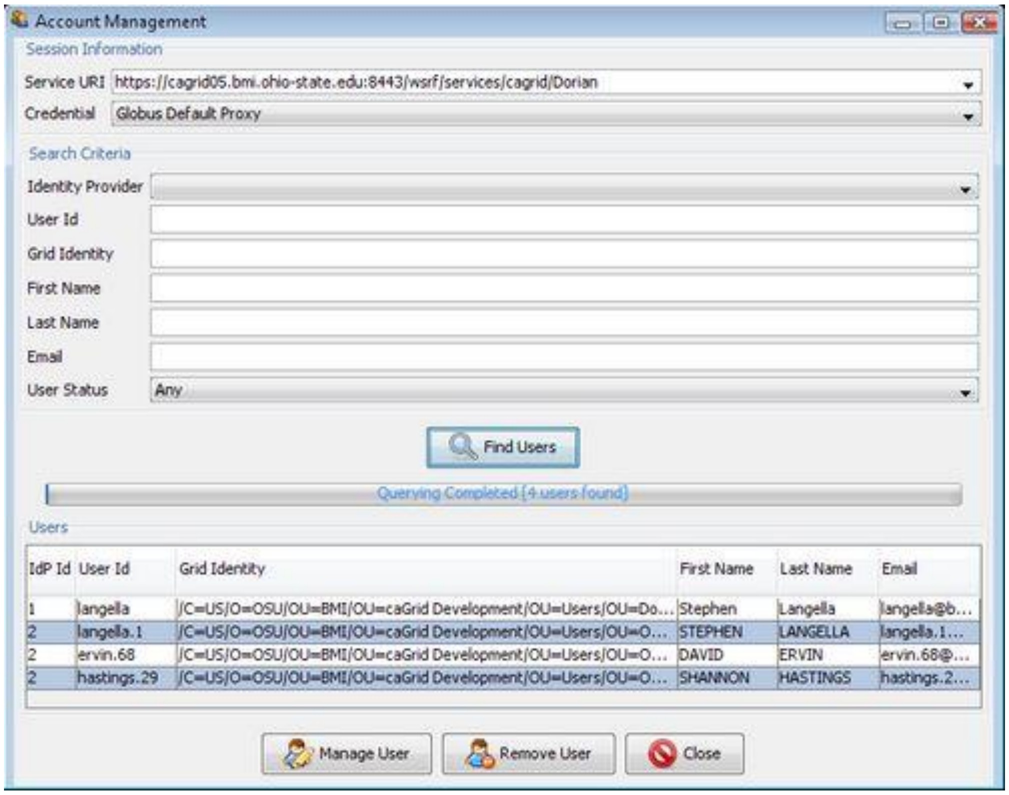
Removing a Trusted Identity Provider

To remove a Trusted IdP, select it from the Trusted Identity Provider Management window and click **Remove Trusted IdP**. Removing a Trusted IdP removes all user accounts associated with the IdP, revoking access to all users associated with the IdP.

Grid User Account Management

Managing grid users and provisioning grid user accounts is the ultimate goal of Dorian. Grid user accounts are created the first time the user attempts to create a grid proxy with a SAML Assertion signed by a Trusted Identity Provider. For each user Dorian maintains a local user id within their IdP, the user's first name, the user's last name, and the user's email address. The information is obtained from the SAML Assertion presented to Dorian when creating a proxy. When a user account is created, Dorian creates a long term certificate and private key for the user, the user's certificate is signed by Dorian's certificate authority. Dorian maintains the user's private key and certificate locally and never distributes it to anyone. Dorian uses the user's private key and certificate in creating and signing grid proxies, in which the user will use to authenticate to grid services. The subject of the user's certificate is composed of 1) Information from Dorian's CA subject, 2) The id of the user's IdP, and 3) the user's local id within the IdP, giving each user a unique identity in the grid. Each user account also has an associated status: Active, Suspended, Pending, or Expired. Only users with an Active status are allowed access to the grid. When a grid user account is first created, the initial status of the account depends on the user policy configured with the user's IdP. If a manual approval policy is specified, the initial status of the grid user account will be Pending, if an automatic approval policy is specified, the initial status of the grid user account is Active. When a user's long term certificate expires, the status of the user's account is set to Expired if the user's IdP specifies a manual renewal policy. In this case an administrator will have to manually renew the user's credentials to grant the user access to the grid again. If however an auto renewal policy is specified for the user's IdP, Dorian automatically renews the user's long term certificate and private key, and the user's account status remains Active. As mentioned earlier, users whose account access is not Active will not be able to create grid proxies; they will also be published in the Dorian Certificate Authority's Certificate Revocation List (CRL), which is published by Dorian to the Grid Trust Service (GTS). Finally each user account is assigned a role within Dorian, either User or Administrator. Users with the Administrator role may create grid proxies; administrate Trusted IdPs, and grid user accounts within Dorian. Users with the User role may only create grid proxies. Use the following steps to administrate grid user accounts using the GAARDS UI.

1. Open the Account Management window by selecting **User Management > Grid Account Management > Grid User Management** *(*Figure 6-11*)*.
2. From this window, you can search for grid user accounts managed by Dorian, manage user accounts, and remove user accounts. To list all grid user accounts managed by a Dorian, select the URI of the Dorian you are interested in from the **Service** drop down menu. If the URI of the Dorian you are interested in is not listed, enter it.
3. Select the grid proxy to use from the **Proxy** drop down menu. Select a proxy of a Dorian administrator.
4. Finally, click **Find Users** to list all the grid user accounts managed by the selected Dorian. To narrow your search, specify search criteria. Dorian supports the following search criteria on grid user accounts: Identity Provider, user id, grid identity, first name, last name, email, and user status. For example, to search for all the accounts that are pending administrative approval, select Pending from the User Status drop down.



Service URI: https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian
Credential: Globus Default Proxy
User Status: Any

Querying Completed (4 users found)

IdP Id	User Id	Grid Identity	First Name	Last Name	Email
1	langella	/C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=Do...	Stephen	Langella	langella@b...
2	langella.1	/C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=O...	STEPHEN	LANGELLA	langella.1...
2	ervin.68	/C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=O...	DAVID	ERVIN	ervin.68@...
2	hastings.29	/C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=O...	SHANNON	HASTINGS	hastings.2...

Figure 6-11 Account Management window

User Management

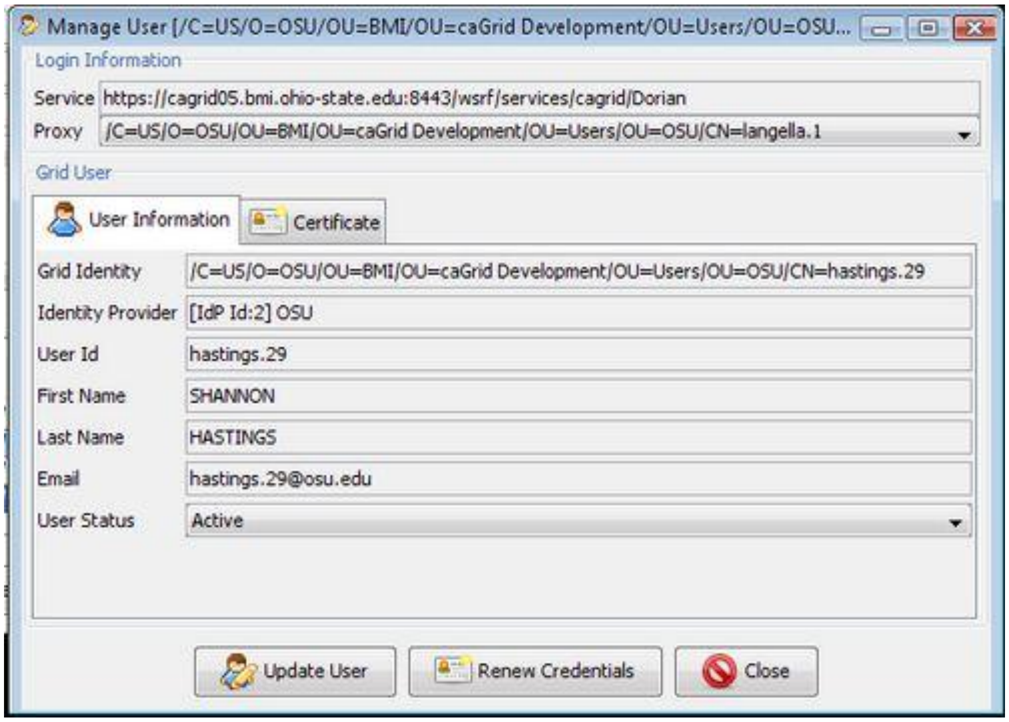
Use the following steps to manage individual grid user accounts through the GAARDS UI.

1. From the **Account Management** window, select the user of interest and click **Manage User**.
2. From the Manage User window (Figure 6-12), you can view the user's information or

change a user’s account status. For example, in the case that the user’s IdP requires manual approval you may change the status from Pending to Active. To revoke a user’s access to the grid, change the user’s account status to Suspended.

3. To commit changes made to a user’s status, click **Update User**, which reflects the changes immediately.

Alternatively, you may renew a user’s long term certificate and private key. You may want to do this if they have expired or if they are going to expire. Details on the user’s long term certificate can be found in the Certificate tab. To renew a user’s long term certificate and private key, click **Renew Credentials**.


Manage User [/C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=OSU...

Login Information
Service: https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian
Proxy: /C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=OSU/CN=langella.1

Grid User
Grid Identity: /C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=OSU/CN=hastings.29
Identity Provider: [IdP Id:2] OSU
User Id: hastings.29
First Name: SHANNON
Last Name: HASTINGS
Email: hastings.29@osu.edu
User Status: Active

Figure 6-12 Manage User window

Removing a Grid User Account

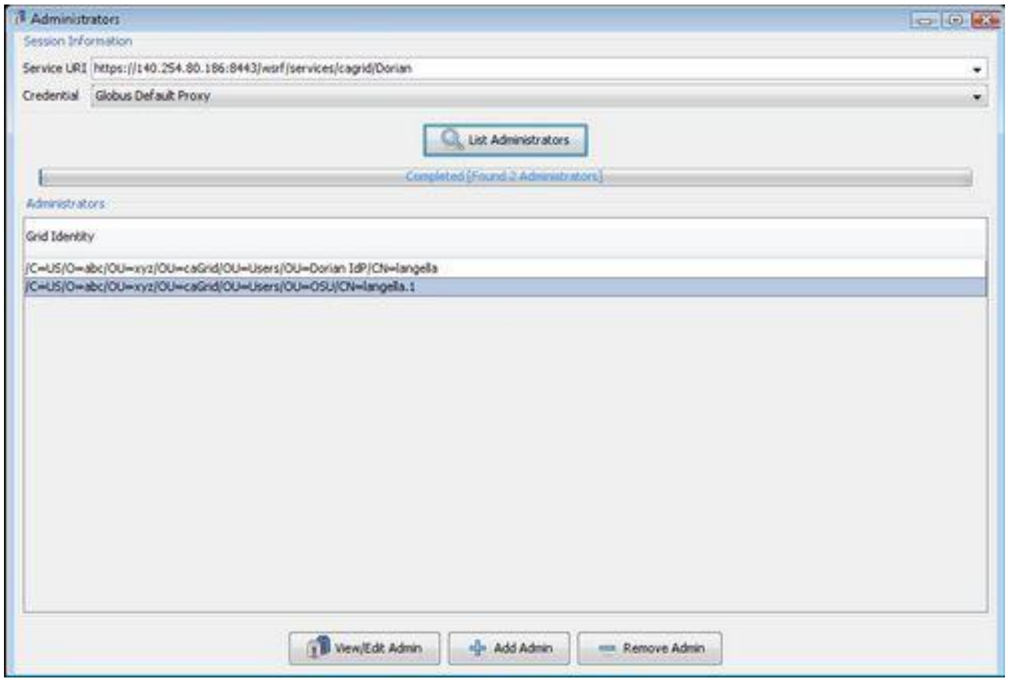
To remove individual grid user accounts through the GAARDS UI, open the Account Management window, select the user to remove, and click **Remove User**. Note that if you remove a grid user account for a user, a new one will automatically be created if they try to create a proxy again. Thus removing an account does not always revoke access to the grid. To disable access to the grid, change the user’s account status to Suspended. In most cases grid user accounts should only be removed if they are no longer affiliated with their Identity Provider.

Managing Administrators

Only users that have been granted administrative access to Dorian will be able to access the administrative features of Dorian. The administrative features include Account Management, Managing Trusted Identity Providers, and the ability to grant administrative access to users. When Dorian is started for the first time, the *dorian* user or the default user has administrative access. The *dorian* user can be used to assign administrative privileges to other users. Note that being granted administrative access does allow the administration of the Local Dorian Identity Provider, which provides a separate mechanism for assigning administrative rights.

Administrative access to Dorian can be managed through the GAARDS UI. To manage administrative access to Dorian using the GAARDS UI, use the following steps.

1. Select **User Management > Grid Account Management > Administrators** from the main menu.
2. In the Administrator window (Figure 6-13), list all the users with administrative access, grant a user administrative access, or revoke a user's administrative access.
3. To list the grid identities of all the entities with administrative access to Dorian, click **List Administrators**. They are listed in the **Administrators** table at the bottom of the screen.
4. To revoke a user's administrative access to Dorian select the user from the Administrators table and click **Remove Admin**.
5. To grant a user administrative access to Dorian click **Add Admin** which opens the Add Administrator window. At the prompt, enter the grid identity of the user you wish to grant administrative access to directly or click **Find** to search for the user.



Service URI https://140.254.80.186:8443/wsrf/services/cagrid/Dorian

Credential Globus Default Proxy

Completed [Found 2 Administrators]

Grid Identity
/C=US/O=abc/OU=xyz/OU=caGrid/OU=Users/OU=Dorian IdP/CN=langella
/C=US/O=abc/OU=xyz/OU=caGrid/OU=Users/OU=OSU/CN=langella.1

Figure 6-13 Administrator window

Host Credential Management

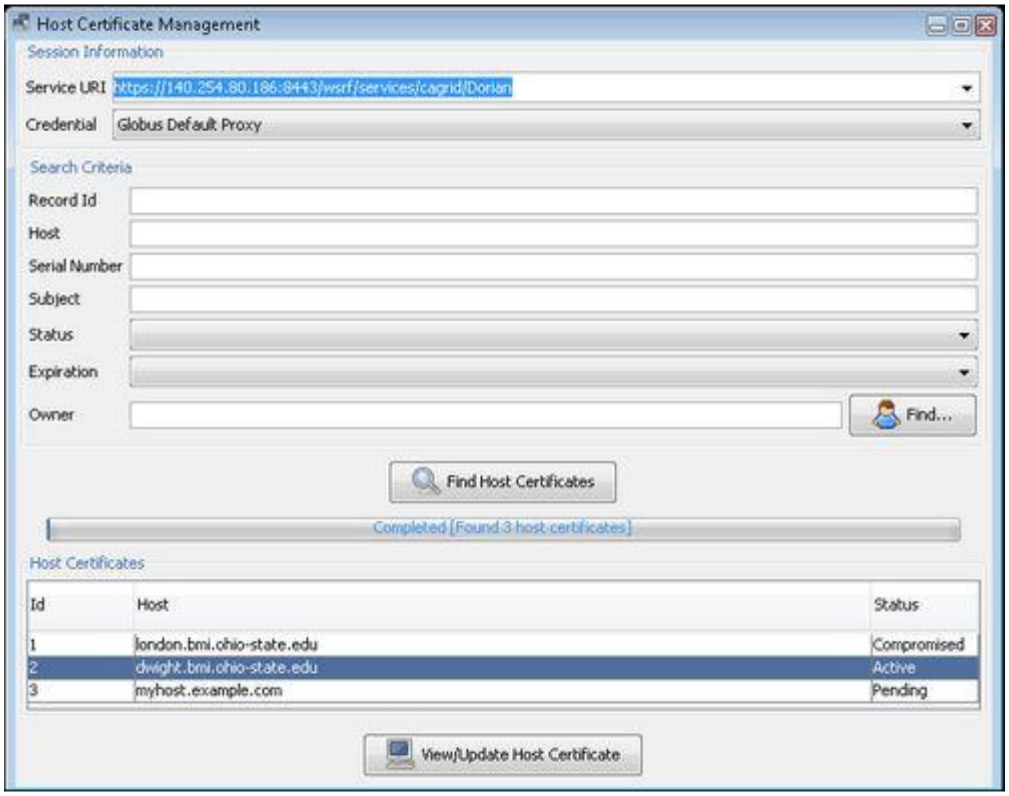
In order to run secure services securely, the container hosting the services must run with a host credential. A host credential consists of an X.509 certificate and private key. Dorian provides a means for users with a grid user account to request a host credential for their services. For each host credential request and each host credential issued Dorian maintains a host credential record. Dorian assigns each host credential record one of the following statuses:

1. **Pending -** Host credentials that have been requested but not yet issued because they require approval of an administrator.
2. **Rejected -** Host credentials that have been requested but were not issued because the request was rejected by an administrator.
3. **Active -** Host credentials that have been issued.
4. **Suspended -** Host credentials that were issued but have been temporarily revoked.
5. **Compromised** - Host credentials that were issued and are permanently revoked.

Host credentials issued by Dorian are bound to a grid user account managed by Dorian. In most cases, a host credential is bound to the user that requested the credential. This binding makes users responsible for any host credentials bound to their account. If a user's account is suspended, any host credentials bound to their account will be revoked and listed in the Dorian CA URL. If a user's account is removed, the status of all the host credentials bound to their account is set to *Compromised*. Each host certificate record is assigned an owner, or the user who the credential is bound to.

The GAARDS UI provides a method of finding/browsing both requested and issued host credentials. To find/browse host credentials use the following steps:

1. Open the Host Certificate Management window by selecting **User Management > Grid Account Management > Host Certificate Management** *(*Figure 6-14*)*.
2. From the **Service URI** drop down menu, select the Dorian you wish to query.
3. From the **Credential** drop down menu, select the grid proxy you wish to use to authenticate with Dorian. Only users with credentials of a Dorian administrator are allowed access to this feature.
4. Click **Find Host Certificates**.


Host Certificate Management
Session Information
Service URI https://140.254.80.186:8443/wsrf/services/cagrid/Dorian
Credential Globus Default Proxy
Search Criteria
Record Id
Host
Serial Number
Subject
Status
Expiration
Owner
Find...
Find Host Certificates
Completed [Found 3 host certificates]
Host Certificates
View/Update Host Certificate

Id	Host	Status
1	london.bmi.ohio-state.edu	Compromised
2	dwight.bmi.ohio-state.edu	Active
3	myhost.example.com	Pending

Figure 6-14 Host Certificate Management window

All the host credential records managed by the Dorian specified are listed. If you wish you may refine your search based on the following criteria:

1. **Record Id** - Dorian assigns each host credential record an id. Specify the record id to pull up a specific record.
2. **Host** - The host name of host.
3. **Serial Number** - The serial number of the certificate issued to the host.
4. **Subject** - The subject of the certificate issued to the host.
5. **Status** - The status of the host credential.
6. **Expiration** - Search based on whether or not the certificate issued to the host has expired. This is useful in determining which host certificates need to be renewed.
7. **Owner** - The grid user that the certificate is bound to.

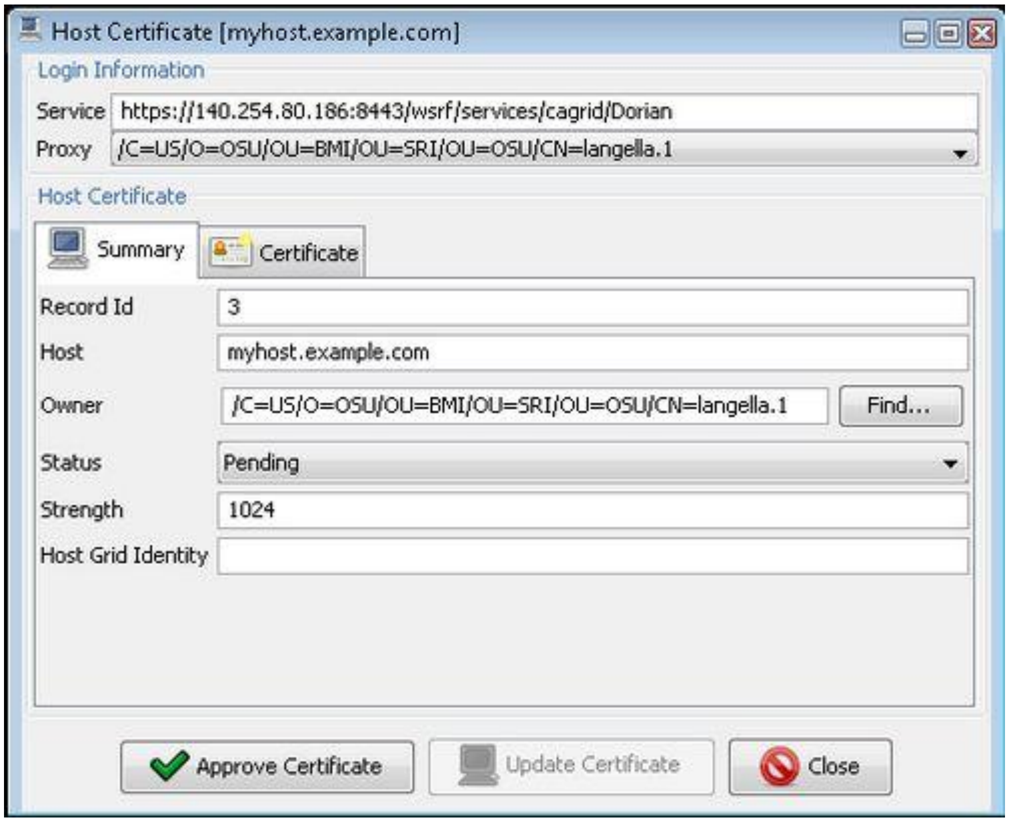
Reviewing Host Credential Requests

Host credentials that require administrative review are assigned a status of *Pending*. It is up to Dorian administrators to decide whether or not to approve a certificate request based on the policy defined for their deployment. The GAARDS UI provides a method for reviewing host credentials requests by using the following steps:

1. From the Host Certificate Management window, select a host credential record with a *Pending* status.

2. Click **View/Update Host Certificate** to open a window containing the details of the host certificate record (Figure 6-15).
3. To approve a host credential request, click **Approve Certificate**. To reject a host credential request, select **Rejected** from the **Status** drop down menu and click **Update Certificate**.

Once the request is reviewed the details of the host certificate record are immediately updated.



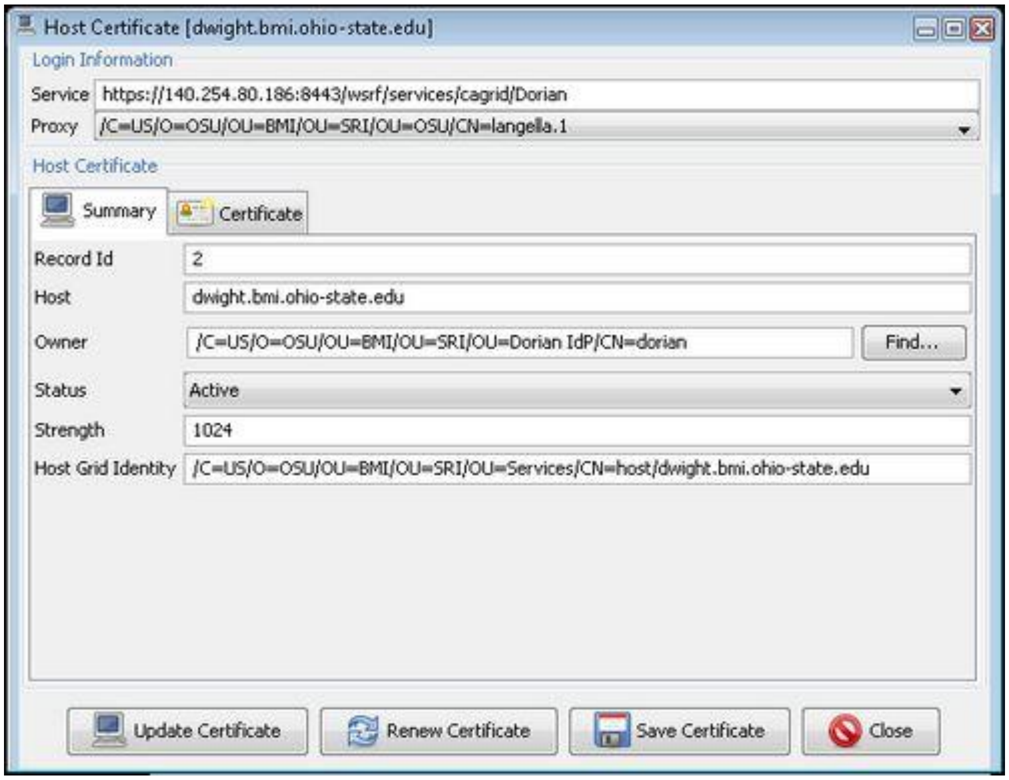
Service	https://140.254.80.186:8443/wsrf/services/cagrid/Dorian
Proxy	/C=US/O=OSU/OU=BMI/OU=SRI/OU=OSU/CN=langella.1
Record Id	3
Host	myhost.example.com
Owner	/C=US/O=OSU/OU=BMI/OU=SRI/OU=OSU/CN=langella.1
Status	Pending
Strength	1024
Host Grid Identity	

Figure 6-15 Host Certificate window

Viewing and Updating Host Credentials

A host credential's status and owner may be modified by a Dorian administrator as long as the current status of the host credential is not *Compromised*. The GAARDS UI provides a method for viewing/updating host credentials requests, by using the following steps:

1. From the Host Certificate Management window, select a host credential record.
2. Click **View/Update Host Certificate** to open a window containing the details of the host certificate record (Figure 6-16).
3. To update a host certificate record, specify the change and click **Update Certificate**.



Service https://140.254.80.186:8443/wsrf/services/cagrid/Dorian
Proxy /C=US/O=OSU/OU=BMI/OU=SRI/OU=OSU/CN=langella.1

Record Id 2
Host dwight.bmi.ohio-state.edu
Owner /C=US/O=OSU/OU=BMI/OU=SRI/OU=Dorian IdP/CN=dorian
Status Active
Strength 1024
Host Grid Identity /C=US/O=OSU/OU=BMI/OU=SRI/OU=Services/CN=host/dwight.bmi.ohio-state.edu

Figure 6-16 Viewing/updating host credentials

Renewing Host Credentials

When Dorian issues host credentials it issues them for the amount of time specified in the *lifetime* element in the Dorian configuration file. After that time the host credentials expire and must be renewed by a Dorian administrator. The GAARDS UI provides a method for renewing host credentials by using the following steps:

1. From the Host Certificate Management window, select a host credential record. (To list all the expired host credentials set the value of the **Expiration** drop down menu to **true** and click **Find Host Certificates**).
2. Click **View/Update Host Certificate** to open a window containing the details of the host certificate record.
3. To renew a host credential request click **Renew Certificate**.

Local Dorian Identity Provider

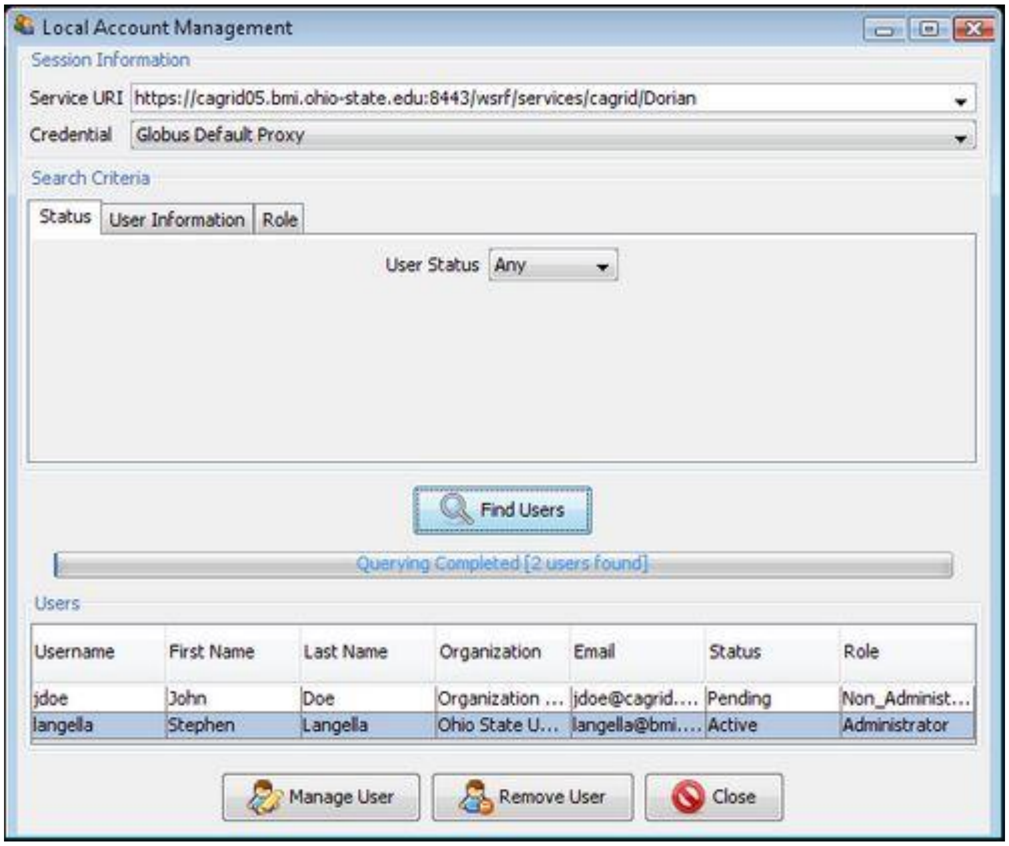
It is anticipated that most users will use their existing locally provided credentials for obtaining grid credentials and only users that are unaffiliated with an existing credential provider should register directly with Dorian. The Dorian Identity Provider (DorianIdP) gives developers, smaller groups, research labs, unaffiliated users, and other groups that don't have their own IdP, the ability to leverage Dorian. The DorianIdP provides a method for prospective users to register for an account. When users register, they create a user id and password which they can subsequently use to authenticate with the Dorian IdP. When a user authenticates, the Dorian

IdP provides the user with a SAML assertion, which can then be used to authenticate with Dorian's to create grid proxies. The DorianIdP provides mechanisms for administrators to manage users; this includes modifying user information (name, address, email, etc.), changing passwords, granting and revoking access, and other administrative actions. All operations provided by the Dorian IdP are made available through Dorian's grid service interface. Administrative operations require administrators to authenticate with a trusted grid proxy. The GAARDS UI also provides a mechanism for Dorian IdP administrators to administrate Dorian IdP user accounts.

Local Account Management

To manage local Dorian IdP account through the GAARDS UI, use the following steps:

1. Select **User Management > Local Account > Local Account Management**.
2. From the Local Account Management window, search for local user accounts managed by the Dorian IdP, manage user accounts, and remove user accounts (Figure 6-17). To list all local user accounts managed by a Dorian IdP, select the URI of the Dorian you are interested in from the **Service** drop down menu. If you do not see the URI of the Dorian you are interested in, enter it.
3. Select the grid proxy to use from the **Proxy** drop down menu. Select a proxy of a Dorian IdP administrator.
4. Click **Find Users** to list all the local user accounts managed by the selected Dorian IdP. To narrow a search, specify search criteria. The Dorian IdP supports the following search criteria on local user accounts: by status, by role, and by user information (first name, last names, address, etc). For example, if you want to search for all the accounts that are pending administrative approval select Pending from the User Status drop down menu.



Service URI: https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian
Credential: Globus Default Proxy
User Status: Any
Querying Completed [2 users found]

Username	First Name	Last Name	Organization	Email	Status	Role
jdoe	John	Doe	Organization ...	jdoe@cagrid....	Pending	Non_Administ...
langella	Stephen	Langella	Ohio State U...	langella@bmi....	Active	Administrator

Figure 6-17 Local Account Management window

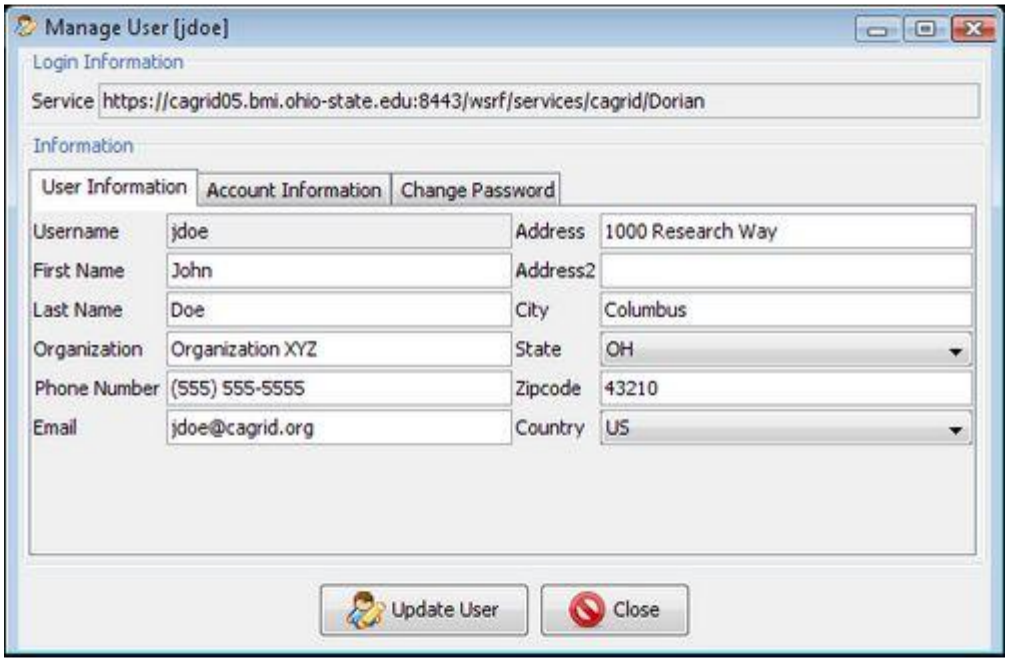
To manage individual local user accounts through the GAARDS UI, use the following steps:

1. From the Local Account Management window select the user of interest and click **Manage User** to open the Manage User window (Figure 6-18).
2. Change the user's demographic information, which includes their first name, last name, mailing address, organization, phone number, and email address. A user's demographic information can also be changed in the **User Information** tab. Through the **Account Information** tab you can also change a user's account information. A user account information consists of their status within the Dorian IdP (Active, Pending, Suspended, Rejected) and the user's role within the Dorian IdP (Administrator or NonAdministrator). Newly registered users may have an account status of Pending meaning an administrator has yet to approve their account. An account can be approved by changing a user's Pending status to Active. Likewise an account can be rejected by changing a user's status from Pending to Rejected. An account can be temporarily suspended or permanently suspended by changing a user's status from Active to Suspended. A temporary account suspension can be removed by changing a user's status from Suspended to Active.

   Note: A User's Status within the Dorian IdP has no relationship to a Dorian grid user account status. Thus having an account in the Dorian IdP does not guarantee that you will have a working grid user account; this depends on the user policy configured for the Dorian IdP within the Identity Federation component of Dorian. Likewise a user's role with the Dorian IdP has no relationship to a user's role with the Identity Federation component of Dorian. Although a Dorian IdP user with an Administrator role in the Dorian IdP may administrate local user accounts in the Dorian IdP, they may not administer grid user accounts.

Finally you may also change a user's account password through the Change Password tab.

3. To commit any changes made to a user's Dorian IdP account, click the *Update User* button; the changes are reflected immediately.



Service https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian

Username	jdoe	Address	1000 Research Way
First Name	John	Address2	
Last Name	Doe	City	Columbus
Organization	Organization XYZ	State	OH
Phone Number	(555) 555-5555	Zipcode	43210
Email	jdoe@cagrid.org	Country	US

Figure 6-18 Manage user window

Grid Trust Service (GTS)

As grid computing technologies gain acceptance and adoption, the transition from highly specialized grids with only a few institutional participants to a grid environment with hundreds of institutions is becoming a reality. Security is of primary importance in the grid and the support for secure communication, authentication, and authorization is a critical requirement, specifically in settings where sensitive data (e.g., patient medical information) must be accessed and exchanged. Also needed are mechanisms to establish and manage "trust" in the grid so that asserted identities and privileges can be verified and validated with the required level of confidence. Within collaboration, it is clear that different institutions have tiered levels of confidence in the users and service management policies of various other institutions. While generally all institutions want to collaborate in some fashion, they have services with varying security policy enforcement requirements. The interconnections between clients and services that are able to securely communicate in the larger grid, form conceptual overlays of trust, which are herein referred to as the "trust fabric" of the grid. Figure 6-19 shows an example trust fabric composed of four trust groups (Trust Groups A-D), over a worldwide grid. The establishment, provisioning, and management of the trust fabric are critical to the scalability, maintenance and security of the grid and other web service environments.

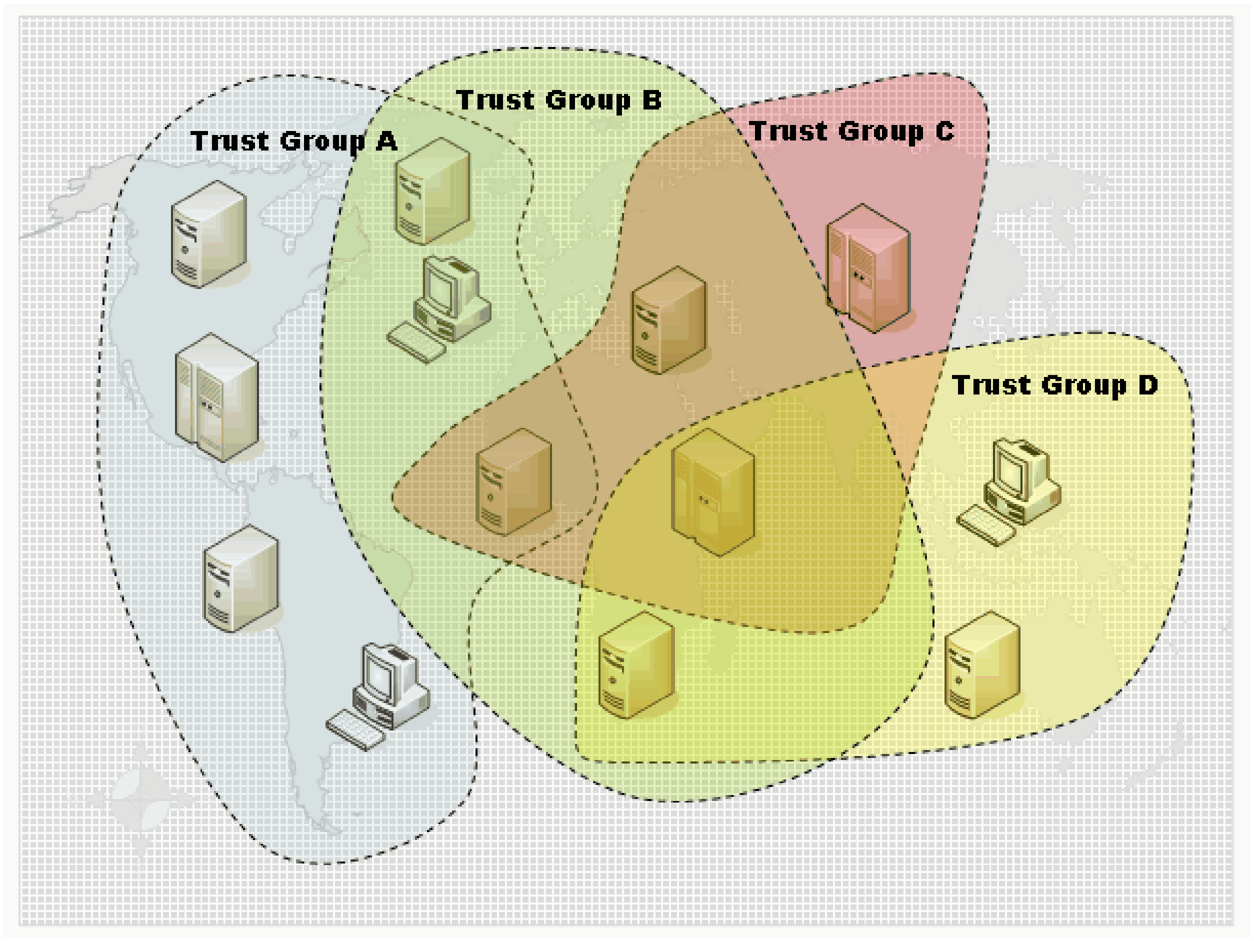


Figure 6-19 Example Grid Trust Fabric

Many components of the grid rely on having trust agreements in place. For example, when a user wants to access a service, they are authenticated based on an identity assigned to them. In the grid, clients and services authenticate with one another using X.509 identity certificates. Grid Identities are assigned to users by authorities. When a grid identity is asserted by an authority in the form of an X.509 identity certificate, it is digitally signed by that authority. Relying parties make authentication decisions based on whether or not the certificate presented is signed by a trusted certificate authority (CA). Thus, authentication requires a trust agreement between the consumers of X.509 identity certificates and the certificate authorities that issue them.

In a grid environment, there may exist tens or even hundreds of certificate authorities, each issuing hundreds if not thousands of certificates. To further complicate the situation, in a dynamic multi-institutional environment, the status of identities may be updated frequently. Identities and credentials can be revoked, suspended, reinstated, or new identities can be created. In addition, the list of trusted authorities may change. In such settings, certificate authorities frequently publish Certificate Revocation Lists (CRL), which specify "black listed" certificates that the authority once issued but no longer accredits. For the security and integrity of the grid, it is critical to be able to perform authentication and validate a given identity against the most up-to-date information about the list of trusted certificate authorities and their corresponding CRLs.

Each institution normally manages its own security infrastructure with its own CAs, and all clients and services within such an administrative domain need to be configured to trust the local trust roots. If collaborations span administrative domains, then participating entities have to be configured to trust the trust roots defined in the different organizations within the limits of their own local policies. The required trust root configurations to participate in such Virtual Organizations (VO) are complex, error prone, and security-policy sensitive. By centralizing the configuration management and provisioning collaborating clients and services “on demand”, one can ensure that the correct and up-to-date trust-root information is made available. In this scenario, the central provisioning server becomes a trusted entity itself, and clients need to be configured to trust its provisioning information. In order to facilitate the trust in the provisioning servers, they should be locally known to the clients, which requires local provision servers to aggregate and to front-end remote ones.

The Grid Trust Service (GTS) is a Web Services Resource Framework compliant federated infrastructure enabling the provisioning and management of a grid trust fabric. The salient features of the GTS can be summarized as follows:

- It provides a complete grid-enabled federated solution for registering and managing certificate authority certificates and CRLs, facilitating the enforcement of the most recent trust agreements.
- It allows the definition and management of trust levels, such that certificate authorities may be grouped and discovered by the level of trust that is acceptable to the consumer.
- The federated nature of the GTS, coupled with its ability to create and manage arbitrary arrangements of authorities into trust levels, allows it to facilitate the curation of numerous independent trust overlays across the same physical grid.
- The GTS can also perform validation for a client, allowing a client to submit a certificate and trust requirements in exchange for a validation decision, which allows for a centralized certificate verification and validation.

This section discusses the administration of the Grid Trust Service (GTS) and only provides a brief overview of the GTS. For more information on the GTS, see the GTS Design Document.

GTS and the Globus Toolkit

The Globus Toolkit implements support for security via its Grid Security Infrastructure (GSI). GSI utilizes X.509 Identity Certificates for identifying a user. An X.509 certificate with its corresponding private key constitutes a unique credential or so-called “Grid credential” that is used to authenticate both users and services within the grid. Under the current Globus release (4.0.3), the authentication process ensures that the X.509 Identity provided by the peer was issued by a trusted certificate authority (CA). However, one limiting issue with the current mechanisms is that trusted CAs and their CRLs are maintained locally on the file system of each Globus installation. When a client authenticates with a service, Globus locates the root CA and CRL of the client’s Identity Certificate on the local file system. Once located, the Globus runtime validates the Identity Certificate against the CA certificate and CRLs. Although this approach is effective, it is difficult to provision CA certificates and CRLs in a large multi-institutional environment, as one has to ensure that all CA and CRL information must be copied to every installation and kept current with the dynamically changing environment. The GTS solves this problem by providing a Grid Service framework for creating, managing, and provisioning of a federated Grid trust fabric. Through its service interface, the GTS provides the ability to register and manage certificate authorities. Using the GTS, Grid entities (services and clients) can discover the certificate authorities in the environment, decide whether or not to trust a certificate authority, and determine the levels of trust assigned to a certificate authority.

Figure 6-20 illustrates how the GTS can be used to enable the Globus Toolkit to authenticate users against the latest trusted certificate authorities. To accomplish this, the GTS provides a framework called SyncGTS, which is embedded in the Globus runtime to automatically synchronize the local trust certificate store with the latest trust fabric maintained in the GTS. Figure 6-20 illustrates how authentication and certificate validation can be performed by leveraging the SyncGTS framework. When a Grid service is invoked, Globus authenticates the client by validating that the Grid proxy provided is signed by a trusted certificate authority. The certificate is validated against a local store as illustrated in the figure. In Figure 6-20, the Dorian certificate authority has been registered with the GTS as a trusted certificate authority and Globus has been configured to synchronize its local trusted certificate store with the GTS. Thus when the OSU user invokes a Grid service using her Dorian-obtained proxy, she will be successfully authenticated by Globus.

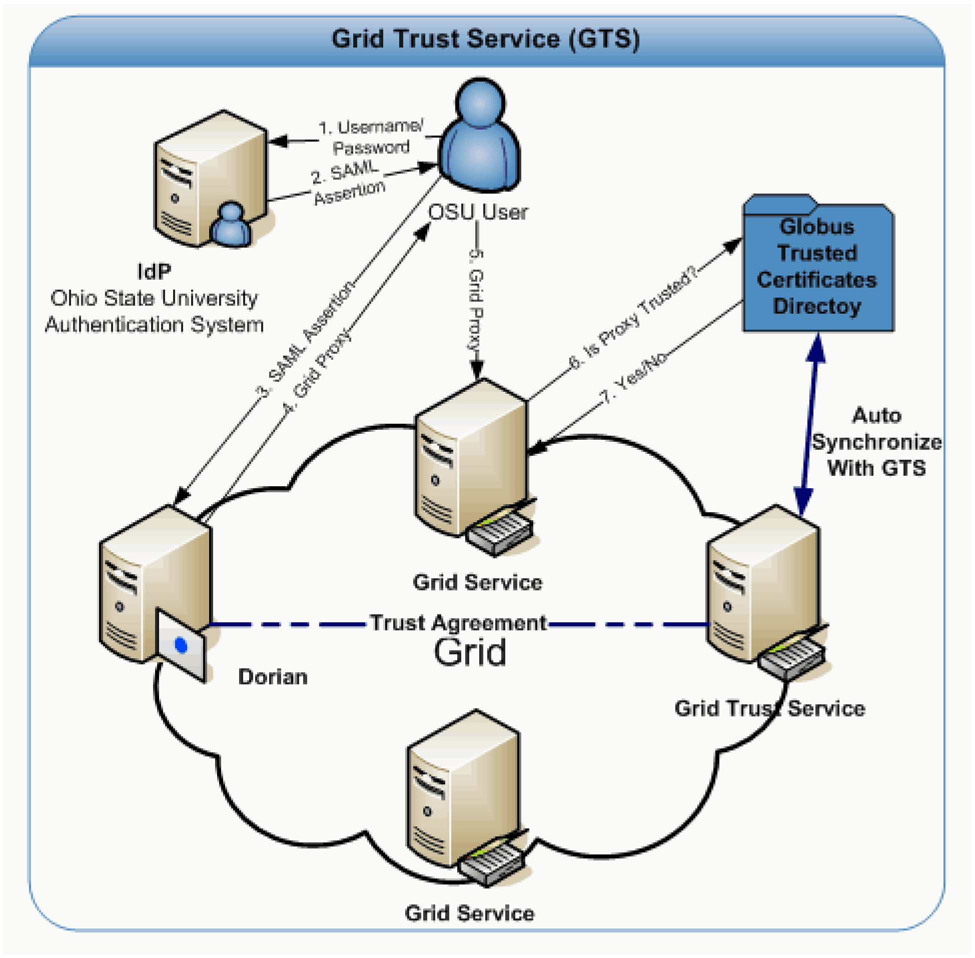


Figure 6-20 GTS Integration with Globus

Installation and Configuration

The Grid Trust Service (GTS) is distributed as standalone project as well as part of other projects such as caGrid. Each of the distributions contains a gts directory herein referred to as GTS_LOCATION. To install and configure the GTS, use the following the steps.

Step 1: Installation Software Prerequisites

Table 6-2 lists the software prerequisites for GTS.

Software	Version	Description
Java SDK	jsdk1.5 or higher	The GTS is written in Java and requires the Java SDK. After installing, set up an environmental variable pointing to the Java SDK directory and name it JAVA_HOME.
MySQL	Mysql 4.1.x or higher	For persisting the trust fabric and other information.

Software	Version	Description
Ant	Ant 1.6.5	The GTS service along with the Globus Toolkit in which the GTS is built on, uses Jakarta Ant for building and deploying.
Globus	Globus 4.0.3	The GTS is built on top of the Globus Toolkit. The GTS requires the ws-core installation of the Globus Toolkit.
Tomcat (Only required if deploying to Tomcat)	Tomcat 5.0.28	The GTS can be optionally deployed as a Grid Service to a Tomcat deployed Globus Toolkit.

Table 6-2 GTS Software Prerequisites

Step 2: Building the GTS

If you have obtained a source release of the GTS, you will need to build the GTS. To build the GTS type the following from a command prompt:

```
%> cd GTS_LOCATION
%> ant clean all
```

Note: Depending on the GTS distribution it may be required to build the entire project that the GTS is distributed with prior to building the GTS. For example, if you obtained a caGrid source distribution this is required. If you received the GTS standalone distribution this is not required.

Step 3: Obtain a Host Credential

Deployments leveraging the GTS to maintain the trust fabric are effectively delegating their authentication responsibility to the GTS. Therefore it is imperative the GTS instance(s) can be trusted. In order for the GTS to be trusted it must run securely with a host credential (X.509 certificate and private key). It is critical that this host credential be issued by an authority that the entities in the deployment trust. If you already have a host credential or have a means of obtaining one please do so and proceed to the next step; otherwise, for the purposes of this guide, it will be shown how to create a certificate authority and use it to issue a host credential. To create a certificate authority, use the following steps from a command prompt (illustrated below):

1. cd GTS_LOCATION
2. Type *ant generateCA*
3. Enter the distinguished name (DN) for the CA (i.e O=xyz,OU=abc,CN=My CA).
4. Enter the number of days that the CA will be valid for (i.e 3650)
5. Enter a password which will be used to encrypt the CA's private key.
6. Enter a file to write the CA private key to.

7. Enter a file to write the CA certificate to.

```
%> cd GTS_LOCATION
%> ant generateCA
Buildfile: build.xml

setGlobus:

checkGlobus:
     [echo] Globus: C:\ext\ws-core-4.0.3

generateCA:
    [input] Please enter the DN for the new CA (ex.
O=xyz,OU=abc,CN=My CA):
O=xyz,OU=abc,CN=My CA
    [input] Please enter the number of days the new CA will be
valid for:
3650
    [input] Please enter a password for the new CA:
password
    [input] Please enter a location to write the new CA's private
key:
cakey.pem
    [input] Please enter a location to write the new CA's
certificate:
cacert.pem
     [java] Successfully create the CA certificate:
     [java] O=xyz,OU=abc,CN=My CA
     [java] CA Certificate Valid Till:
     [java] Fri Jun 23 12:47:10 EDT 2017
     [java] CA Private Key Written to:
     [java] cakey.pem
     [java] CA Certificate Written to:
     [java] cacert.pem

BUILD SUCCESSFUL
Total time: 46 seconds
```

Once a certificate authority is created, use it to issue host credentials. To create host credentials, use the following steps from a command prompt (illustrated below):

1. cd GTS_LOCATION
2. Type *ant createAndSignHostCertificate*
3. Enter the location of the CA's private key.
4. Enter the password used to encrypt the CA's private key.
5. Enter the location of the CA's certificate.
6. Enter the name of the host.
7. Enter the number of days that the host credentials should be valid for.
8. Enter a location to write the host private key.

9. Enter a location to write the host certificate.

```
%> cd GTS_LOCATION
%> ant createAndSignHostCertificate
Buildfile: build.xml

setGlobus:

checkGlobus:
     [echo] Globus: C:\ext\ws-core-4.0.3

createAndSignHostCertificate:
    [input] Please enter the location of the CA's private key:
cakey.pem
    [input] Please enter the CA's password:
password
    [input] Please enter the location of the CA's certificate:
cacert.pem
    [input] Please enter the Hostname [${env.HOST}]:
myhost
    [input] Please enter the number of days the host certificate
will be valid f
or:
365
    [input] Please enter a location to write the host key:
hostkey.pem
    [input] Please enter a location to write the host certificate:
hostcert.pem
     [java] Successfully create the user certificate:
     [java] O=xyz,OU=abc,CN=host/myhost
     [java] User certificate issued by:
     [java] O=xyz,OU=abc,CN=My CA
     [java] User Certificate Valid Till:
     [java] Wed Jun 25 13:58:37 EDT 2008
     [java] User Private Key Written to:
     [java] hostkey.pem
     [java] User Certificate Written to:
     [java] hostcert.pem

BUILD SUCCESSFUL
Total time: 52 seconds
```

Step 4: Configure Globus To Trust the GTS

In order for the GTS to be used to distribute trust roots Globus must be configured to trust the CA that issued the host credentials obtained in the previous step. Place a copy of the certificate for the CA that issued the host credentials in the Globus trusted certificates directory. Unless otherwise specified during installation, the Globus trusted certificate directory is usually *USER_HOME/.globus/certificates*. Globus requires all CA certificates in its trusted certificates directory to be in PEM format and to have a digit extension (0-9). For example, if a CA certificate is stored in the file *cacert.pem* in PEM format than in order to configure Globus to trust this certificate authority it should be copied to the directory *USER_HOME/.globus/certificates* (create directory if needed) with the file name *cacert.0*

Step 5: Configuring a Secure Container

Now that you have obtained host credentials, you may use them to configure a secure container. The GTS can be run from a secure Globus container or a secure Tomcat container. For directions on how to configure a secure Globus container consult the following web site: http://www.cagrid.org/mwiki/index.php?title=CaGrid:How-To:SecureGlobusContainer For directions on how to configure a secure Tomcat container consult the following website: http://www.cagrid.org/mwiki/index.php?title=CaGrid:ConfigureTomcat

Step 6: Configuring the GTS

The GTS is configured through a single configuration file which is located at *GTS_LOCATION/etc/gts-conf.xml* (shown below). The GTS uses a MySQL Database as its backend data store; you must provide the GTS with the connection details for your MySQL database. The *database* element in the GTS configuration is used to specify the connection information for your MySQL database. In the majority of cases you only need to specify the *hostname* of your database server, the *port* that the server runs on, and the *username* and *password* of a database user. When the GTS is first initialized it creates a database, named with the value of the gts-internal-id element. The GTS also proceeds to setup its database schema in the database it created. In order to do so the GTS must be configured with a database user that has the appropriate permissions. If you do not wish to provide the GTS with such a user you may create the database manually and provide the GTS with a user who has the permission to modify the database schema. In this scenario the GTS will not create the database but will proceed to setup its database schema in the database that was manually created.

```
<gts>
   <resource name="GTSConfiguration"
class="gov.nih.nci.cagrid.gts.service.GTSConfiguration">
       <gts-config>
           <gts-internal-id>GTS</gts-internal-id>
           <sync-authorities hours="0" minutes="2" seconds="0"/>
           <database>
               <name/>
               <driver>com.mysql.jdbc.Driver</driver>
               <urlPrefix>jdbc:mysql:</urlPrefix>
               <host>localhost</host>
               <port>3306</port>
               <username>root</username>
               <password></password>
               <pool>1</pool>
            </database>
        </gts-config>
    </resource>
</gts>
```

Step 7: Set Initial Administrators

Many of the operations provided by the GTS provide a means of administrating the trust fabric and are therefore restricted to GTS administrators. GTS Administrators are *"super users"* and can perform any operation on a GTS (i.e., manage certificate authorities, manage trust levels, manage permissions, etc). In order to bootstrap the GTS such that it may be administered through its service interface the GTS must be provided with at least one initial administrator. The GTS provides a command line program for adding initial administrators. To leverage this program, type the following from a command prompt:

```
%> cd GTS_LOCATION
%> ant addAmin
```

This prompts for the grid identity of the initial administrator to add. Enter the grid identity of the user you want to add as an initial administrator as shown below:

```
%> cd GTS_LOCATION
%> ant addAmin
Buildfile: build.xml

setGlobus:

checkGlobus:
     [echo] Globus: C:\ext\ws-core-4.0.3

addAdmin:
    [input] Please enter the grid identity for the admin you wish
to add:
/C=US/O=OSU/OU=BMI/OU=caGrid Development/OU=Users/OU=Dorian
IdP/CN=jdoe
     [java] The user /C=US/O=OSU/OU=BMI/OU=caGrid
Development/OU=Users/OU=Dorian
 IdP/CN=jdoe was succesfully added as an administrator of the GTS
(GTS)

BUILD SUCCESSFUL
Total time: 48 seconds
```

Step 8: Deploying the GTS

Once you have configured a secure container (Globus or Tomcat) you need to deploy the GTS to that container. To deploy the GTS to a secure Globus container type the following from a command prompt:

```
%> cd GTS_LOCATION
%> ant deployGlobus
```

To deploy the GTS to a secure Tomcat container type the following from a command prompt:

```
%> cd GTS_LOCATION
%> ant deployTomcat
```

Regardless of which container is selected, a significant amount of output displays on the screen. If the deployment is successful, *"BUILD SUCCESSFUL"* displays on the screen.

Step 9: Verifying the Installation

Once GTS is deployed, the installation and configuration of GTS is complete. Before verifying that the installation was successful, start the GTS service by starting the container that the GTS was deployed to. For directions on starting a secure Globus container consult the following web site: http://www.cagrid.org/mwiki/index.php?title=CaGrid:How-To:SecureGlobusContainer

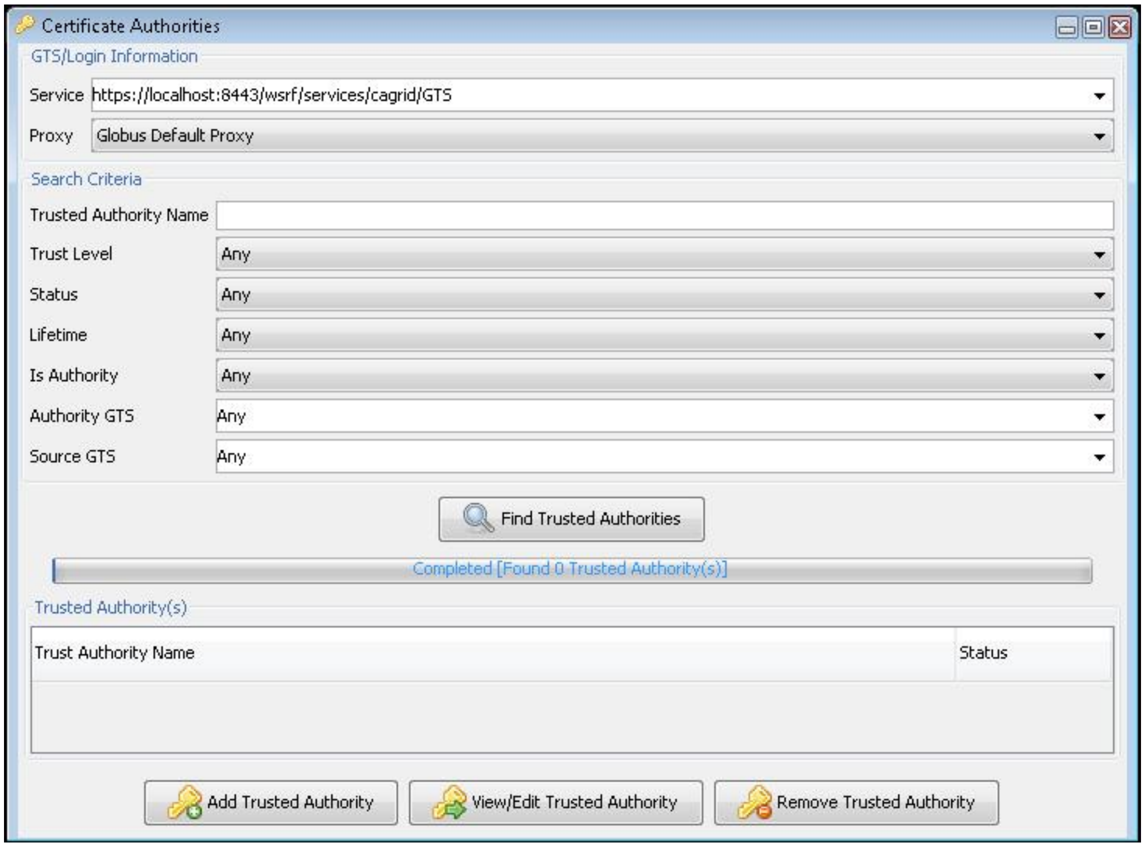
To start a secure Tomcat container run the startup script (`startup.sh` or `startup.bat`) located in TOMCAT_INSTALLATION_DIRECTORY/bin. If the container starts up, verify that the GTS installation was successful by typing the following from the command prompt:

```
%> cd GTS_LOCATION
%> ant ui
```

After the GTS Administration UI opens, use the following steps:

1. Click **Certificate Authorities** to open the Certificate Authorities window.
2. From the **Service** drop down menu, select *https://localhost:8443/wsrf/services/cagrid/GTS*
3. Click **Find Trusted Authorities**.

The UI connects to the GTS and gets a list of all the trusted certificate authorities managed by that GTS. Since this is a new installation this should return 0 certificate authorities and should be reflected in the status message displayed in the progress bar below the Find Trusted Authorities button. Upon successfully connecting to the GTS, the value of the status message in the progress bar should update to *Completed [Found 0 Trusted Authority(s)]* as illustrated in Figure 6-21. The Grid Trust Service (GTS) has been successfully installed and configured.



Service https://localhost:8443/wsrf/services/cagrid/GTS
Proxy Globus Default Proxy

Trusted Authority Name
Trust Level Any
Status Any
Lifetime Any
Is Authority Any
Authority GTS Any
Source GTS Any

Completed [Found 0 Trusted Authority(s)]

Figure 6-21 Certificate Authorities window

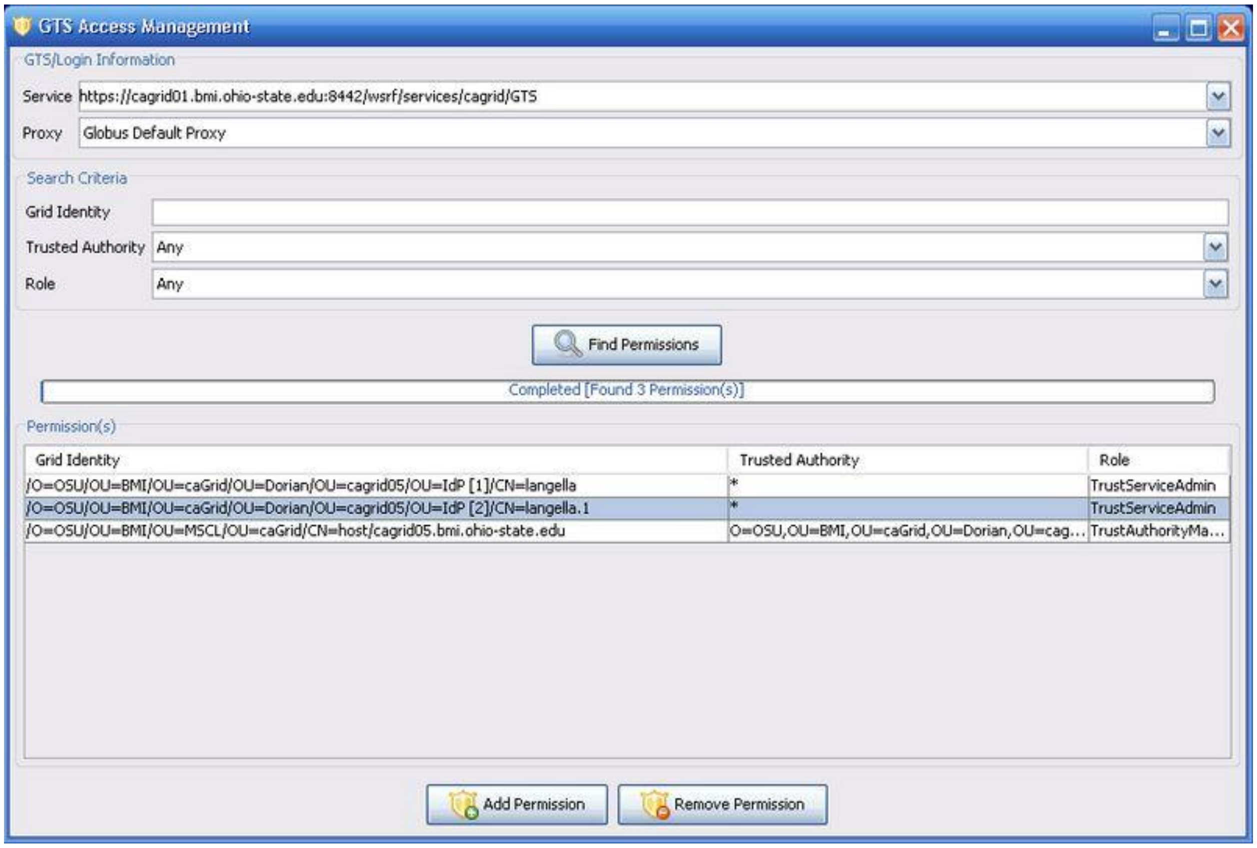
GTS Administration

Managing Trust Fabric Administrators

Many of the operations provided by the GTS provide a means of administrating the trust fabric and are therefore restricted to GTS administrators or to administrators of individual certificate authorities. The GTS allows for the assignment of two types of permissions: GTS Administrators and Trusted CA Administrators. GTS Administrators are *"super users"* and can perform any operation on a GTS (i.e., manage certificate authorities, manage trust levels, manage permissions, etc). Trusted CA Administrator permission corresponds to a specific CA giving a user with this permission the ability to update the CRL for the corresponding CA. The GAARDS UI enables the remote management of GTS administrators. To manage administrators with the GAARDS UI, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric > Permissions** to open the Permissions window.
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **List Permissions**.

The permissions granting administrative access to the selected GTS are listed. Figure 6-22 lists three permissions. The first two give the entity with the grid identity listed super user rights to the GTS. The third permission gives the entity with the grid identity rights to administrate the Trusted Certificate Authority listed, mainly the ability to update the Certificate Revocation List (CRL).



Service: https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Proxy: Globus Default Proxy

Grid Identity:
Trusted Authority: Any
Role: Any

Completed [Found 3 Permission(s)]

Grid Identity	Trusted Authority	Role
/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella	*	TrustServiceAdmin
/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1	*	TrustServiceAdmin
/O=OSU/OU=BMI/OU=MSCL/OU=caGrid/CN=host/cagrid05.bmi.ohio-state.edu	O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cag…	TrustAuthorityMa…

Figure 6-22 GTS Access Management window

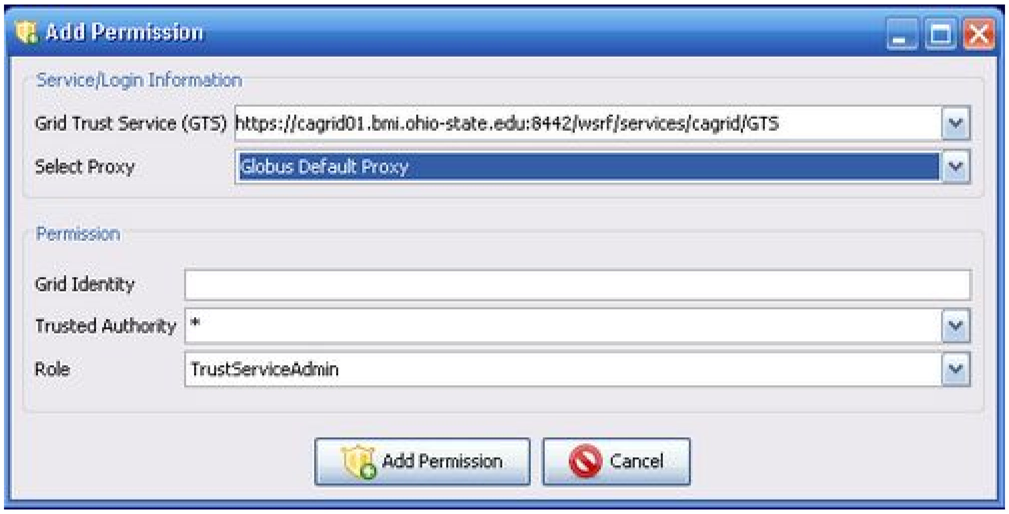
Adding a GTS Administrator

Administrators or permissions can be added by using the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric > Permissions** to open the Permissions window.
4. Click **Add Permission** to launch the Add Permission window (Figure 6-23).
5. Select the URI of the GTS in which you want to add an administrator.
6. In the **Grid Identity** text box, enter the grid identity of the user being added as an administrator.
7. From the **Trusted Authority** drop down menu, select the trusted certificate authority in

which this permission will apply. Selecting "*" or all trusted authorities makes the permission apply to all trusted certificate authorities, giving the entity specified super user right to the GTS. Selecting a specific Trusted Certificate Authority gives the grid identity specified the right to update the CRL for the certificate authority selected.

8. Click **Add Permission** to add/apply the permission to the GTS.



Grid Trust Service (GTS): https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Select Proxy: Globus Default Proxy
Grid Identity:
Trusted Authority: *
Role: TrustServiceAdmin

Figure 6-23 Add Permission window

Removing a GTS Administrator

To remove an administrator, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric > Permissions** to open the Permissions window.
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **List Permissions**.
6. Select the permission to remove.
7. Click **Remove Permission**.

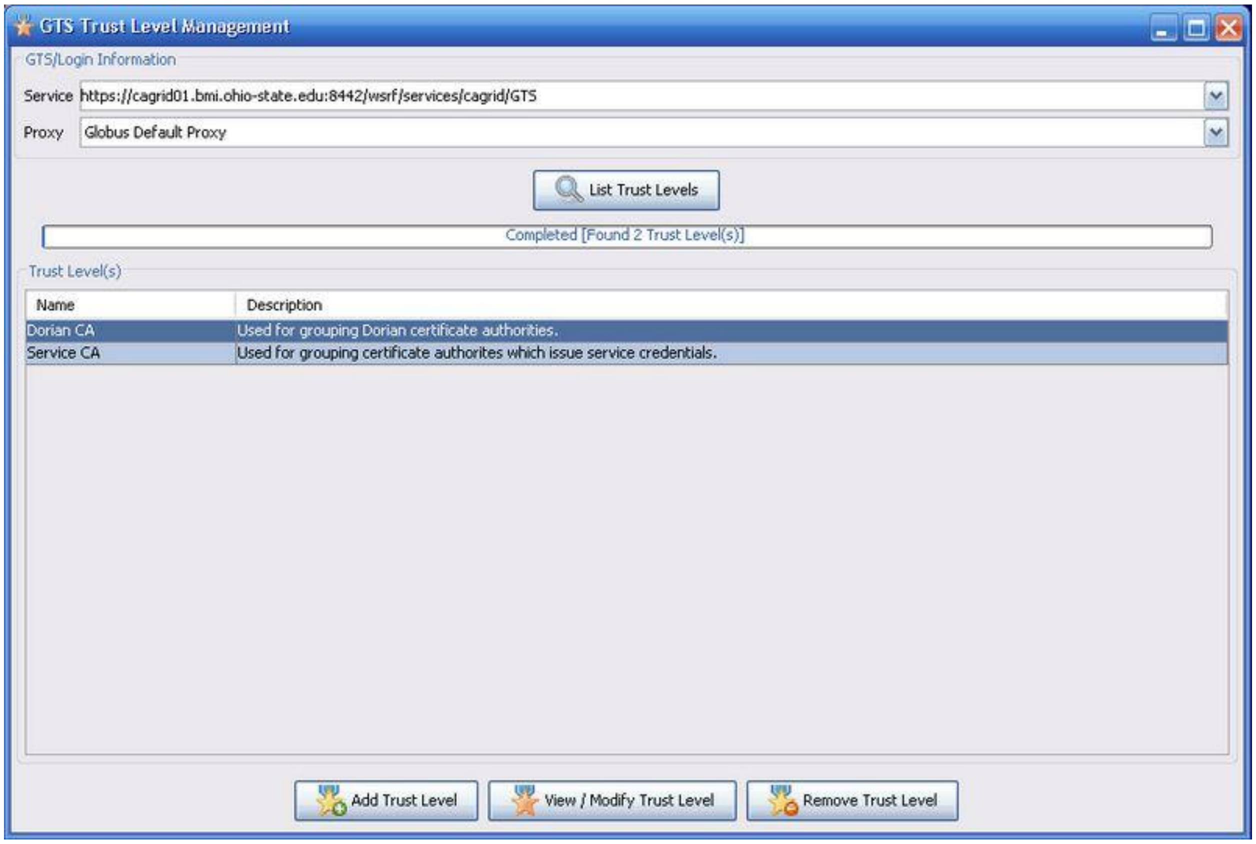
Managing Levels of Assurance

A level of assurance or trust level specifies the level of confidence with which a given certificate authority is trusted in the grid in which it is deployed. The trust level concept in the grid is similar to obtaining an identification card, for example obtaining a passport requires extensive documentation and a thorough background check where as obtaining a library card requires much less. In the grid, one can assume that certificate authorities are trusted with different levels of confidence. There are multiple types and instances of certificate authorities. Some authorities may be used to assert identities; other authorities may be used to assert digitally signed documents. Even certificate authorities asserting the same thing may have differing levels of trust associated with them, as they may employ different policies for issuing and validating identities. For example, a certificate authority may require that anyone applying for a certificate present official documentation about their real identity. The CA issues a certificate to the applicant after these documents are reviewed by the CA staff. Another certificate authority may automatically issue certificates based on an online application submitted by the applicant; the applicant may have been requested to log on to the system using a user id and password. In these cases, the first certificate authority has a stricter policy for issuing certificates; thus, it is reasonable to expect that the first certificate authority should be trusted more than the second certificate authority

The GAARDS UI enables the remote management of trust levels by using the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric >Levels of Assurance** to open the Levels of Assurance window (Figure 6-24).
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **List Trust Levels**.

All the trust levels or levels of assurance registered with the selected GTS are listed. This UI also allows GTS administrators to add, remove, and update trust levels.



Service https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Proxy Globus Default Proxy

Completed [Found 2 Trust Level(s)]

Name	Description
Dorian CA	Used for grouping Dorian certificate authorities.
Service CA	Used for grouping certificate authorites which issue service credentials.

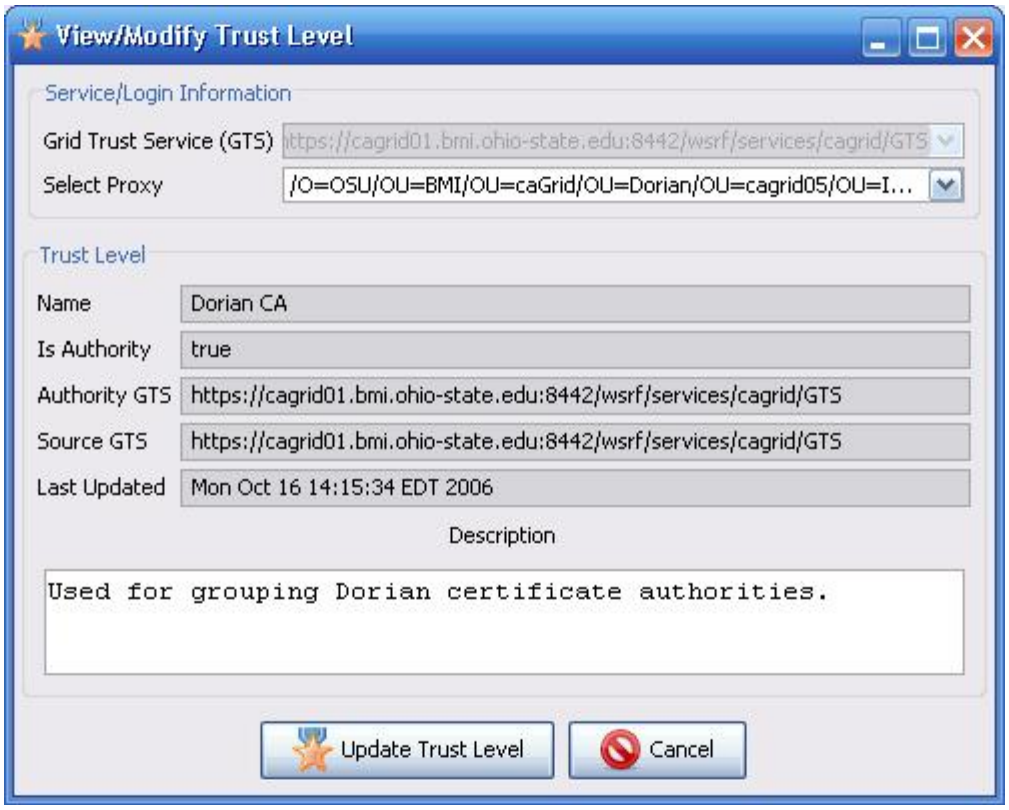
Figure 6-24 GTS Trust Level Management

Viewing and Modifying Levels of Assurance

To view or modify a trust level, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric > Levels of Assurance** to open the Levels of Assurance window (Figure 6-25).
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **List Trust Levels**.
6. Select the trust level in which you wish to view or modify.
7. Click **View/Modify Trust Level** to open the View/Modify window for the trust level you selected.

The View/Modify window lists the name of the trust level, whether or not the selected GTS is the authority for the trust level, the authority GTS, the source GTS, when the trust level was last updated, and a description of the trust level. Since the trust fabric can be federated, a GTS can inherit trust levels from other GTSs. The GTS in which a trust level originates is considered the authority GTS. The Is Authority listing in Figure 6-25 specifies whether the selected GTS is the authority for the trust level. The Authority GTS listing specifies the URL of the authority GTS or the GTS in which the trust level originated. The Source GTS listing specifies the URL of the GTS in which the selected GTS discovered or obtained the trust level from. The *Description* listing in the figure specifies a human readable description of the trust level. The *Description* is the only field of a trust level that the GTS allows an administrator to modify. To modify the Description, simply make the desired changes and click **Update Trust Level**.



Grid Trust Service (GTS): https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS

Select Proxy: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=I...

Trust Level

Name	Dorian CA
Is Authority	true
Authority GTS	https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Source GTS	https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Last Updated	Mon Oct 16 14:15:34 EDT 2006

Description

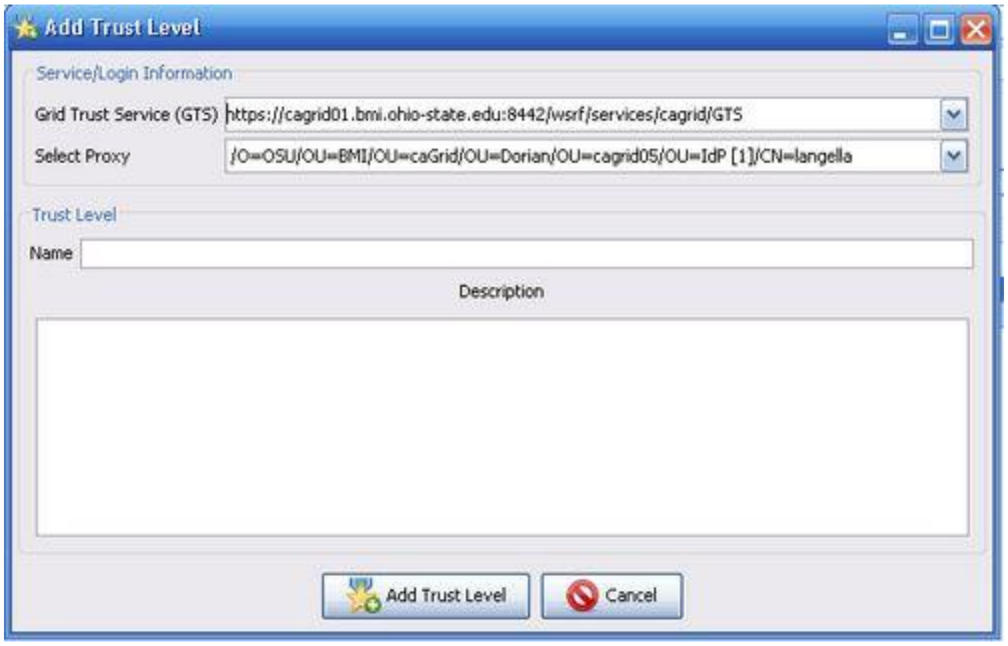
```
Used for grouping Dorian certificate authorities.
```

Figure 6-25 View/Modify Trust Level window

Adding a Level of Assurance

To add a new trust level to a GTS, use the following steps.

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric > Levels of Assurance** to open the Levels of Assurance window.
4. Click **Add Trust Level** to open the Add Trust Level window (Figure 6-26).
5. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
6. Specify the name and description of the trust level.
7. Click **Add Trust Level** to submit the new trust level to the GTS.



Grid Trust Service (GTS) https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS

Select Proxy /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella

Figure 6-26 Add Trust Level window

Removing a Level of Assurance

To remove a trust level, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric > Levels of Assurance** to open the Levels of Assurance window.
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **List Trust Levels**.
6. Select the trust level to remove.
7. Click **Remove Trust Level**.

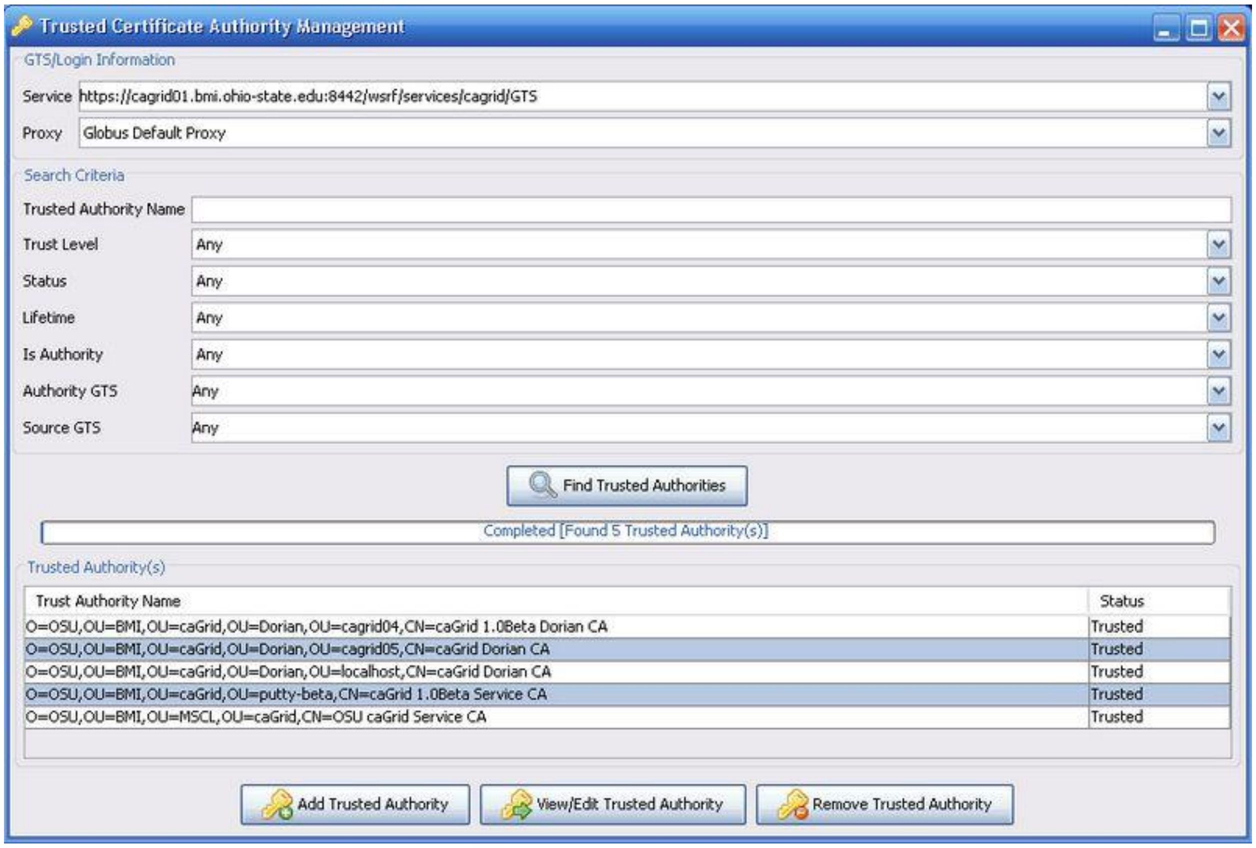
Managing Certificate Authorities

The ultimate goal of the GTS is to provide a framework for provisioning trusted certificate authorities to both clients and services in the grid such that they may confidently know which certificate authorities to trust when deciding whether to accept credentials, assertions, and other digitally signed documents. Thus management of certificate authorities within the GTS or trust fabric is critical. The GAARDS UI facilitates the management of certificate authorities within the trust fabric. Certificate Authorities are managed through the Trusted Certificate Authority Management Window. Although you may browse trusted certificate authorities without being a GTS administrator, you must be a GTS administrator to manage the trusted certificate authorities. Thus depending on what you are planning to do you may be required to provide the grid credentials of a GTS administrator.

To manage certificate authorities with the GAARDS UI use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account.
3. Select **Trust Fabric > Certificate Authorities** to open the Certificate Authorities window (Figure 6-27).
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **Find Certificate Authorities**.

Once the search is complete, the Trusted Certificate Authorities meeting your search criteria is listed in the table below the progress bar. The certificate authorities are listed by name and by the subject distinguished name (DN) in the certificate.



Service: https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Proxy: Globus Default Proxy

Trusted Authority Name:
Trust Level: Any
Status: Any
Lifetime: Any
Is Authority: Any
Authority GTS: Any
Source GTS: Any

Completed [Found 5 Trusted Authority(s)]

Trust Authority Name	Status
O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cagrid04,CN=caGrid 1.0Beta Dorian CA	Trusted
O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cagrid05,CN=caGrid Dorian CA	Trusted
O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=localhost,CN=caGrid Dorian CA	Trusted
O=OSU,OU=BMI,OU=caGrid,OU=putty-beta,CN=caGrid 1.0Beta Service CA	Trusted
O=OSU,OU=BMI,OU=MSCL,OU=caGrid,CN=OSU caGrid Service CA	Trusted

Figure 6-27 Trusted Certificate Authority Management window

Adding a Trusted Certificate Authority

To add or register a certificate authority, the GTS requires the specification of the CA’s root certificate, a set of trust levels, a status, and an optional CRL. The CA’s root certificate is required for validating certificates. The set of trust levels specifies the level of trust associated with the CA. The status specifies the current state of the certificate authority and can be set to “trusted” or “suspended”. Setting the status of a certificate authority allows it to be temporarily added and removed from the trust fabric. For instance, if the security of a CA has been compromised, its status can be set to "suspended" to quickly invalidate all certificates issued and signed by the CA. For each trusted certificate authority, the GTS maintains a Certificate Revocation List (CRL). The CRL contains a list of certificates that have been revoked by the CA. To add a CA to the GTS, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account.
3. Select **Certificate Authorities** to open the Certificate Authorities window.
4. Click **Add Trusted Authority** to open the Add Trusted Authority window (Figure 6-28).
5. From the **Service** drop down menu, select the URI of the GTS you wish to add a CA to.
6. Click **Import Certificate** which prompts for the location of the CA certificate.
7. Browse to the location of the CA certificate (PEM format), select it, and click **Open** to load it.
8. In the **Trust Levels** tab, select all the trust levels that apply to the certificate authority being added.
9. Optionally, if the CA has a CRL, you may import it. Just as the CA certificate, the CRL must be in PEM format. To import the CA's CRL, click **Import CRL**.
10. Click **Add Trusted Authority**.

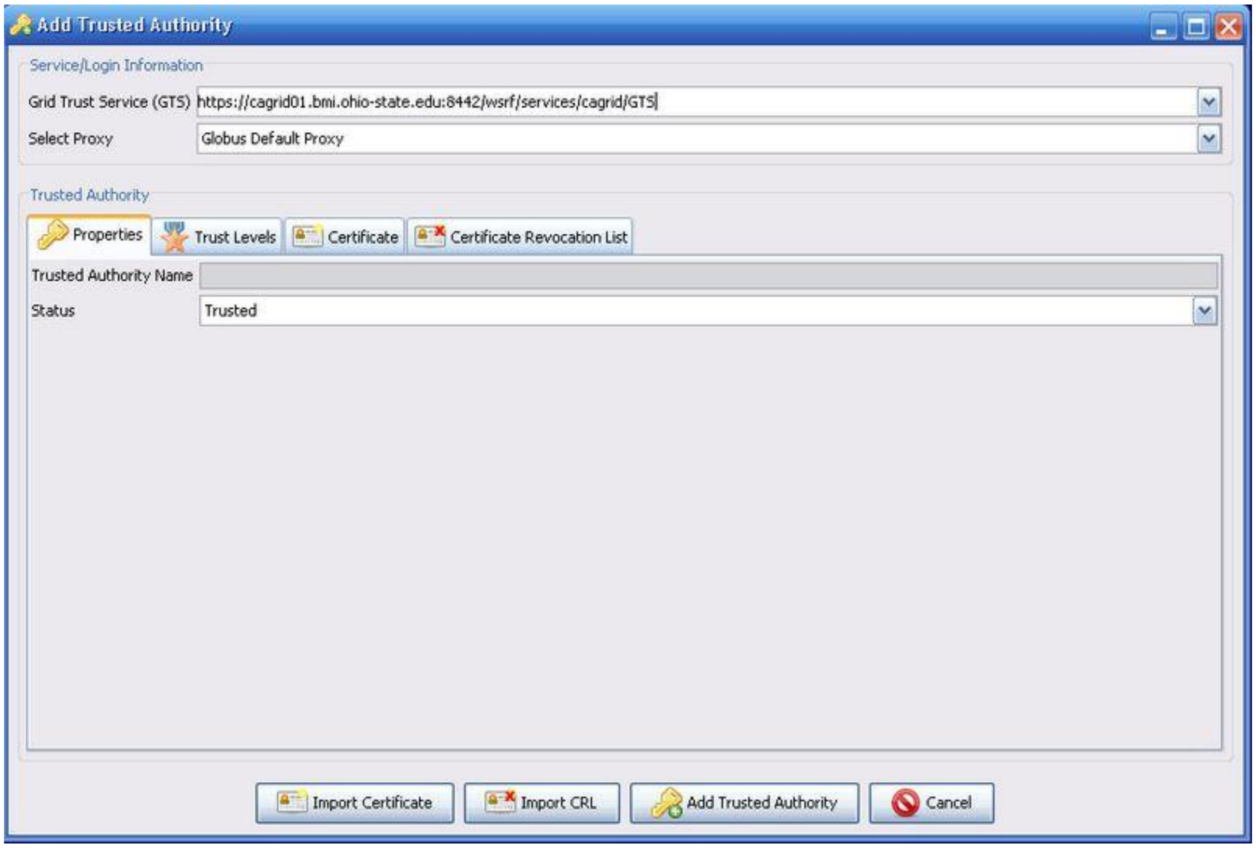

Grid Trust Service (GTS): https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Select Proxy: Globus Default Proxy
Trusted Authority Name:
Status: Trusted

Figure 6-28 Add Trusted Authority window

Viewing/Modifying Trusted Certificate Authorities

To view or modify a Trusted Certificate Authority, use the following steps.

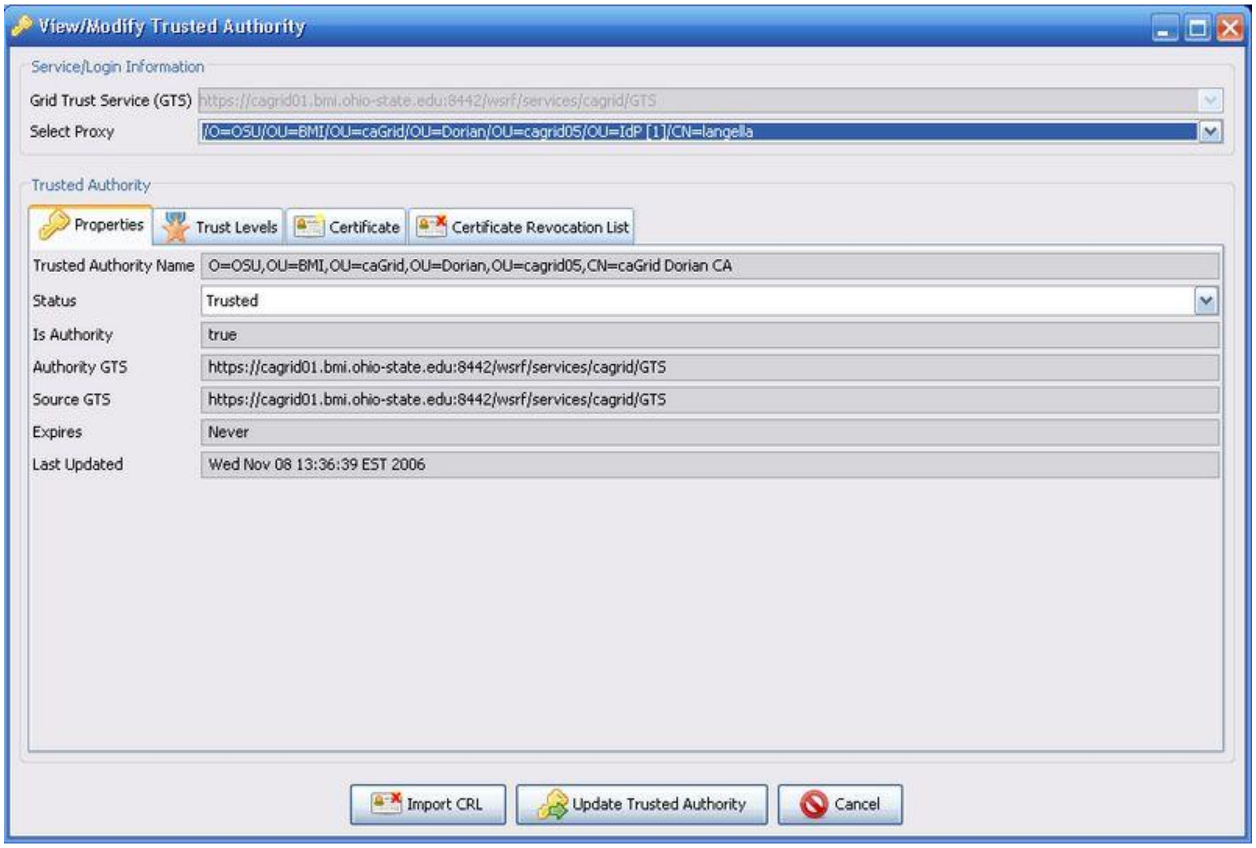
1. Launch the GAARDS UI.
2. Logon to the Grid using your user account.
3. Select **Trust Fabric > Certificate Authorities** to open the Certificate Authorities window.
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **Find Certificate Authorities**.
6. Select the certificate authority to view/modify and click **View/Modify Trusted Authority** to open the View/Modify Trusted Authority window (Figure 6-29).

The View/Modify Trusted Authority window contains four tabs: Properties, Trust Levels, Certificate, and Certificate Revocation List (CRL). The Properties tab contains various metadata about the certificate authority including the Trusted Authority Name, Status, Is Authority, Authority GTS, Source GTS, Expires, and Last Updated. The *Trusted Authority Name* field specifies the subject distinguished name (DN) in the CA's certificate. The *Status* field specifies the current state of the certificate authority; the status can be set to *Trusted* or *Suspended*. Setting the status of a certificate authority allows it to be temporarily added and removed from the trust fabric. For instance, if the security of a CA has been compromised, its status can be set to *Suspended* to quickly invalidate all certificates issued and signed by the CA. The *Status* field can be modified. The remaining fields in the Properties tab correspond to the federated nature of the GTS. The *Is Authority* field specifies whether or not the GTS is the authority for the certificate authority. The *Authority GTS* field specifies the URI of the authority GTS for the certificate authority. The *Source GTS* listing specifies the URI of the GTS from which the GTS discovered or obtained the certificate authority. The *Expires* field specifies when the certificate authority listing within the GTS expires, a certificate authority can expire if it was discovered from another GTS and communication with the other GTS is lost. If the GTS is the authority for the certificate authority, it never expires. The *Last Updated* field specifies the last time the certificate authority listing was updated.

The Trust Levels tab contains a listing of all the trust levels supported by the GTS. The trust levels, in which the certificate authority is assigned, are selected in the listing. The trust levels for the certificate authority can be modified by selecting and deselecting specific trust levels.

The Certificate tab presents a graphical view of the certificate authority's certificate and cannot be modified. The CRL represented in the Certificate Revocation List tab can be modified by clicking the **Import CRL** button, which prompts to specify the location of the CRL on the file system. Note that the CRL must be in PEM format.

To update a certificate authority, specify the desired changes and click **Update Trusted Authority**.



Grid Trust Service (GTS): https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Select Proxy: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella

Trusted Authority Name	O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cagrid05,CN=caGrid Dorian CA
Status	Trusted
Is Authority	true
Authority GTS	https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Source GTS	https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Expires	Never
Last Updated	Wed Nov 08 13:36:39 EST 2006

Figure 6-29 Show/Modify Trusted Authority window

Removing Trusted Certificate Authorities

To remove a Trusted Certificate Authority, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account.
3. Select **Trust Fabric > Certificate Authorities** to open the Certificate Authorities window.
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **Find Certificate Authorities**.
6. Select the certificate authority to remove.
7. Click **Remove Trusted Authority**.

Managing a Federated Grid Trust Fabric

Redundancy and scalability are critical properties of a federated trust fabric. Serious performance implications will occur if all entities in the grid are not discovering and performing validation against a trust fabric maintained in a central GTS. In order to enable a federated trust fabric, each GTS can be administered to synchronize with a set of authoritative GTSs. A GTS can inherit both trust levels and trusted certificate authorities from its authority GTSs.

Registering an authority GTS requires the specification of the following properties:

- a service's uniform resource identifier (URI)
- priority
- whether or not to synchronize the trust levels
- time to live
- whether or not to perform authorization
- the authority service's identity

The priority property is used for resolving conflicts between authority GTSs. For example, if two authority GTSs have a listing for the same certificate authority, the authority GTS with the highest priority is used for obtaining that certificate authority and its corresponding information (e.g. it's CRL). If contact to an authoritative GTS is lost for a significant amount of time, the trust fabric within the subordinate GTS may become significantly out of date, which could be a potential security risk. The time to live property specifies how long certificate authorities obtained from authoritative GTSs are valid in the subordinate GTS. The time to live on a given certificate authority record is reset after each synchronization with the authority GTS. If contact with an authority GTS is lost, the time to live expires and the certificate authority is removed from the subordinate's trust fabric.

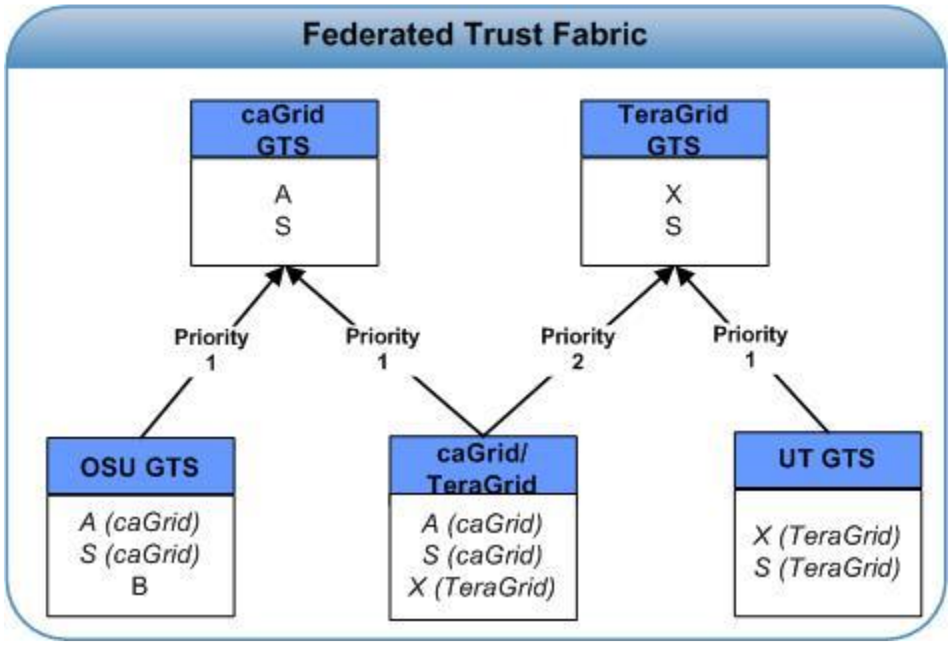


Figure 6-30 Deployment of multiple GTSs

Figure 6-30 illustrates an example of how multiple GTSs can be deployed to create and manage a federated trust fabric. In the example there are five GTSs: caGrid GTS, TeraGrid GTS, OSU GTS, caGrid/TeraGrid GTS, and UT GTS.

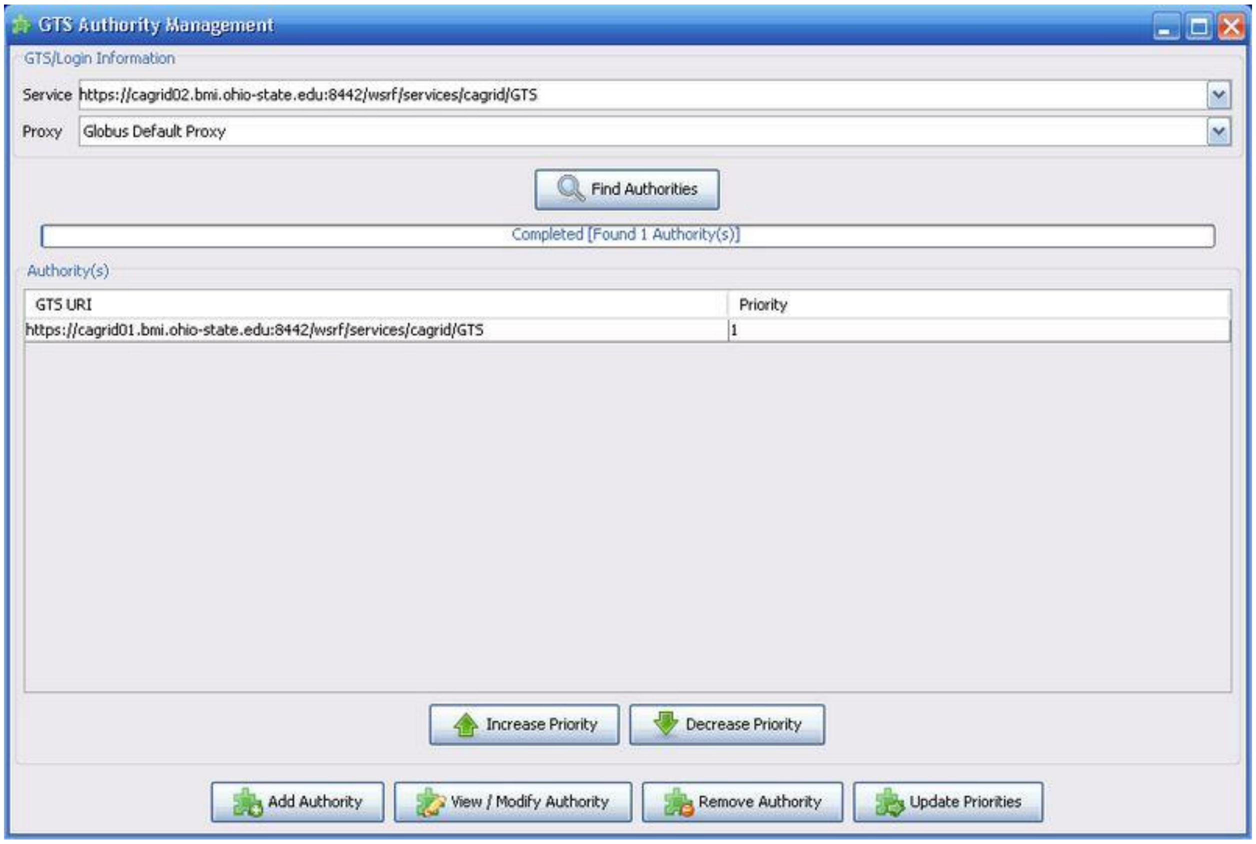
- The caGrid GTS has no authority GTSs; it manages the certificate authorities A and S.

- The TeraGrid GTS has no authority GTSs; it manages the certificate authorities X and S.
- The OSU GTS has one authority GTS, the caGrid GTS. The OSU GTS inherits the certificate authorities A and S from its authority, the caGrid GTS. The OSU GTS manages an additional certificate authority B. The OSU GTS is an example of how the global trust fabric can be extended to include local trusted certificate authorities, in this case, and the additional certificate authority CA B, which is trusted by OSU.
- The caGrid/TeraGrid GTS has two authority GTSs: the caGrid GTS and the TeraGrid GTS. The TeraGrid GTS inherits CA A from the caGrid GTS and CA X from the TeraGrid GTS. Since the caGrid GTS has a higher priority then the TeraGrid GTS, it inherits CA S from the caGrid GTS. The caGrid/TeraGrid GTS is an example of how two existing trust fabrics from two different Grids can be joined together.
- Finally, the UT GTS has one authority GTS, the TeraGrid GTS. The UT GTS inherits CA X and CA S from the TeraGrid GTS. The UT GTS is an example of standing up a GTS for better redundancy and scalability.

The GAARDS UI facilitates the management of a federated trust fabric through the administration of Authority GTSs. A GTS with an Authority GTS inherits its trust levels and trusted certificate authorities. To manage AuthorityGTS(s) with the GAARDS UI, use the following steps:

1. Launch the GAARDS UI.
2. Log on to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric > Trust Federation** to open the Trust Federation window (Figure 6-31).
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **Find Authorities**.

All the Authority GTSs are listed in the table below the progress bar. Each Authority GTS is listed with its service URL and its priority; the priority dictates how conflicts are resolved between authorities. For example, if a GTS has two authorities that manage the same certificate authority, the GTS inherits the certificate authority from the Authority GTS with the higher priority (lowest number).



Service: https://cagrid02.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS

Proxy: Globus Default Proxy

Completed [Found 1 Authority(s)]

GTS URI	Priority
https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS	1

Figure 6-31 GTS Authority Management window

Prioritizing Authorities

To prioritize the Authority GTSs, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric > Trust Federation** to open the Trust Federation window.
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **Find Authorities**.
6. Select the desired Authority GTS to change the priority of.
7. Click **Increase Priority** to increase the priority of the selected Authority GTS or click **Decrease Priority** to decrease the priority of the selected Authority GTS. (The Authority GTS with the lowest number has the highest priority).
8. Once the priorities of the Authorities GTSs are organized properly, click **Update Priorities** to commit the priorities to the GTS.

Adding an Authority GTS

To add an Authority GTS, use the following steps:

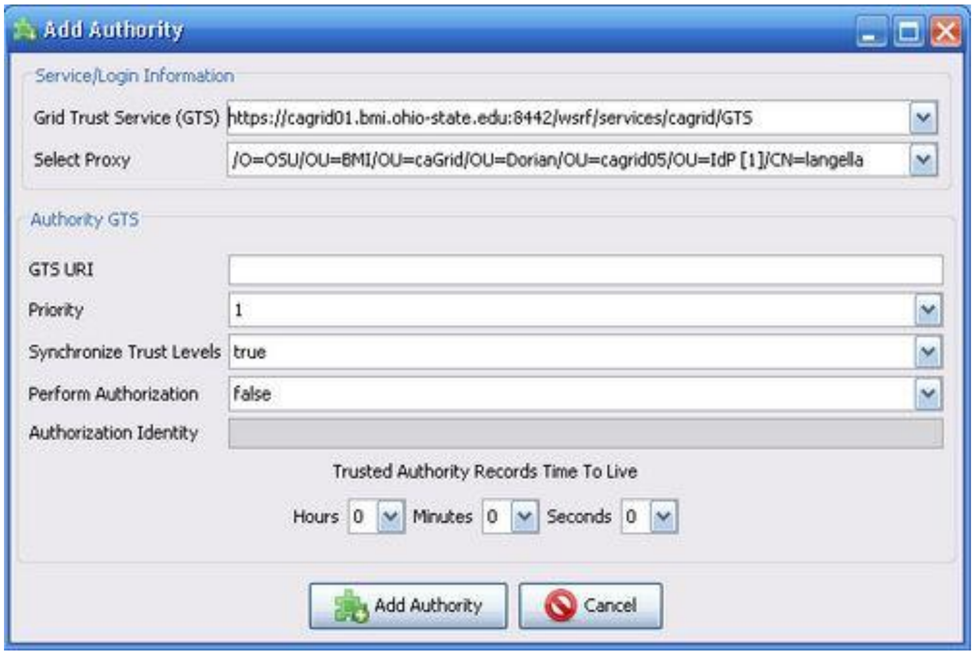
1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric > Trust Federation** to open the Trust Federation window.
4. Click **Add Authority** to open the Add Authority window (Figure 6-32).
5. From the **Grid Trust Service (GTS)** drop down menu, specify the URI of the GTS you wish to add an authority to.

   In the **GTS URI** text field enter the URI of the GTS being added as an authority.

   From the **Priority** drop down menu, specify the priority of this authority with respect to other Authority GTSs.

   From the **Synchronize Trust Levels** drop down menu, specify whether or not the GTS should synchronize or inherit the trust levels from its Authority GTS.

   From the **Perform Authorization** drop down menu, specify whether or not the GTS should perform authorization on the Authority GTS. If performing authorization is chosen, the expected grid identity of the Authority GTS must be entered in the **Authorization Identity** text field.
6. Finally, specify a time to live for trusted authorities inherited from the authority GTS. For example, if the time to live specified is an hour, certificate authorities inherited are removed from the local GTS trust fabric after an hour if contact to the authority GTS is lost.
7. Click **Add Authority**.


Add Authority
Service/Login Information
Authority GTS
Trusted Authority Records Time To Live
Add Authority
Cancel

Grid Trust Service (GTS): https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Select Proxy: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella
GTS URI:
Priority: 1
Synchronize Trust Levels: true
Perform Authorization: false
Authorization Identity:
Hours 0 Minutes 0 Seconds 0

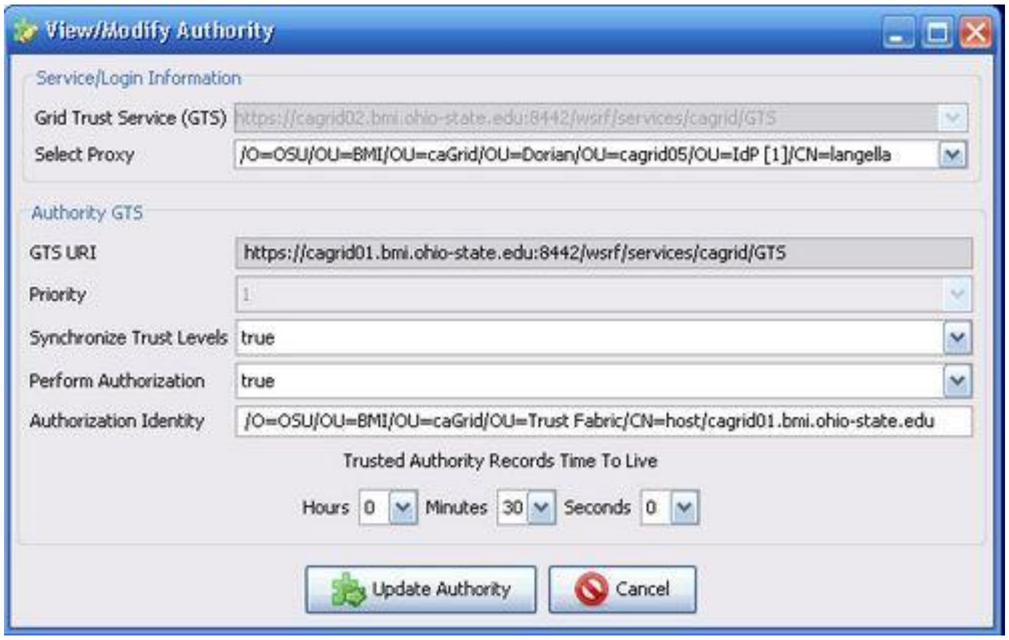
Figure 6-32 Add Authority window

Updating an Authority GTS

To update an Authority GTS, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric > Trust Federation** to open the Trust Federation window.
4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **Find Authorities**.
6. Select the authority you wish to view/update.
7. Click **Update Authority** to open the Update Authority window (Figure 6-33).

You may update whether or not the GTS should synchronize or inherit the trust levels from its Authority GTS. You may also update the authorization constraints for the Authority GTS as well as the time to live for trusted certificate authorities obtained from the authority GTS. You may not update the Authority GTS URI or priority.



Grid Trust Service (GTS): https://cagrid02.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Select Proxy: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella
GTS URI: https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Priority: 1
Synchronize Trust Levels: true
Perform Authorization: true
Authorization Identity: /O=OSU/OU=BMI/OU=caGrid/OU=Trust Fabric/CN=host/cagrid01.bmi.ohio-state.edu
Trusted Authority Records Time To Live
Hours 0 Minutes 30 Seconds 0

Figure 6-33 View/Modify Authority window

Removing an Authority GTS

To remove an Authority GTS, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a GTS Administrator).
3. Select **Trust Fabric >Trust Federation** to open the Trust Federation window.

4. From the **Service** drop down menu, select the URI of the GTS you wish to administer.
5. Click **Find Authorities**.
6. Select the authority you wish to remove.
7. Click **Remove Authority**.

Syncing with the Trust Fabric

The Globus Toolkit facilitates the authentication of clients against a list of trusted certificate authorities. This consists of validating the client's certificate to ensure that it was issued and signed by a trusted certificate authority. The Grid Trust Service (GTS) maintains the trusted certificate authorities or trust fabric for caGrid. In order for Globus to authenticate users against the trust fabric, both client and server caGrid installations must be synced with the trust fabric. The SyncGTS tool provides three methods for syncing with the trust fabric:

1. Command Line Approach
2. Service Based Approach
3. Programmatic Approach

The command line approach is generally recommended for clients wishing to sync with the trust fabric. The Service Based Approach is recommended for syncing servers with the trust fabric. The programmatic approach is intended for applications who wish to sync with trust fabric on behalf of the clients who use them, removing this responsibility from the client.

SyncGTS Command Line Approach

This guide provides a step by step process for syncing with the trust fabric using the SyncGTS Command Line Approach. The SyncGTS command line approach is intended to be used to sync client environments with the trust fabric. To ensure that the client environment is synced with the latest trust roots, this approach should be repeated regularly.

SyncGTS is distributed as a standalone project as well as part of other projects such as caGrid. Each of the distributions contains a syncgts directory herein referred to as SYNC_GTS_LOCATION. To sync with the trust fabric using the command line approach use the following steps:

Step 1: Building SyncGTS

If you have obtained a source release of the SyncGTS you will need to build the SyncGTS. To build the SyncGTS type the following from a command prompt:

```
%> cd SYNC_GTS_LOCATION
%> ant clean all
```

Note: Depending on the SyncGTS distribution it may be required to build the entire project that SyncGTS is distributed with prior to building SyncGTS. For example, if you have obtained a caGrid source distribution this is required. If you received a SyncGTS standalone distribution this is not required.

Step 2: Configuring SyncGTS (Optional)

SyncGTS is configured through an XML configuration file herein referred to as the *Sync Description*. The default *Sync Description* file can be found in *SYNC_GTS_LOCATION/ext/resource/sync-description.xml*. For most distributions, SyncGTS is pre-configured to work with the grid in which it is being distributed. Therefore no further configuration is required. If you do need to make configuration changes to SyncGTS, in most cases you will only need to edit the *gtsServiceURI*, *GTSIdentity*, and *ExcludedCAs* elements. For more details on the SyncGTS configuration file please consult the following website:

http://www.cagrid.org/mwiki/index.php?title=GTS:1.1:Administrators_Guide:SyncGTS:Configuration

Step 3: Installing GTS Trust Roots (Optional)

In order for SyncGTS to sync with a GTS service, it is required that the local environment trust the GTS service it is synced with. In other words the local environment must trust the certificate authority that issued the GTS Service's credentials. Most distributions of SyncGTS are pre-configured to trust the GTS credentialing certificate authority for the grid in which the distribution is configured. If this is the case no further configuration is required. If this is not the case SyncGTS can easily be configured to trust other certificate authorities by placing a copy of the CA's certificate in the directory: *SYNC_GTS_LOCATION/ext/resources/certificates*. The CA certificate must be contained in PEM format and must be given a digit (0-9) extension. For example, to configure SyncGTS to trust a CA whose certificate is contained in the file cacert.pem, the file should be renamed to cacert.0 and copied to the directory: *SYNC_GTS_LOCATION/ext/resources/certificates*. In most cases you will also to add an entry to the excluded CA list in SyncGTS's configuration file. For more details on the SyncGTS configuration file please consult the following website:

http://www.cagrid.org/mwiki/index.php?title=GTS:1.1:Administrators_Guide:SyncGTS:Configuration

Step 4: Running SyncGTS

To run SyncGTS from the command line, type the following from a command prompt:

```
%> cd SYNC_GTS_LOCATION
%> ant syncWithTrustFabric
```

SyncGTS Server Side Approach

This guide provides a step by step process for syncing with the trust fabric using the SyncGTS Server Side Approach. The SyncGTS server side approach is intended to be used to sync server environments or environments running grid services with the trust fabric. In this approach SyncGTS is deployed to a service container ensuring that the server environment is automatically updated to be in sync with the most up to date trust fabric.

SyncGTS is distributed as a standalone project as well as part of other projects such as caGrid.

Each of the distributions contains a syncgts directory herein referred to as SYNC_GTS_LOCATION. To sync with the trust fabric using the server side approach, use the following steps:

Step 1: Building SyncGTS

If you have obtained a source release of the SyncGTS you will need to build the SyncGTS. To build the SyncGTS type the following from a command prompt:

```
%> cd SYNC_GTS_LOCATION
%> ant clean all
```

Note: Depending on the SyncGTS distribution it may be required to build the entire project that SyncGTS is distributed with prior to building SyncGTS. For example if you have obtained a caGrid source distribution this is required. If you received a SyncGTS standalone distribution this is not required.

Step 2: Configuring SyncGTS (Optional)

SyncGTS is configured through an XML configuration file herein referred to as the *Sync Description*. The default *Sync Description* file can be found in*SYNC_GTS_LOCATION/ext/resource/sync-description.xml*. For most distributions, SyncGTS is pre-configured to work with the grid in which it is being distributed. Therefore no further configuration is required. If you do need to make configuration changes to SyncGTS, in most cases you will only need to edit the *gtsServiceURI*, *GTSIdentity*, and *ExcludedCAs* elements. For more details on the SyncGTS configuration file please consult the following website:

http://www.cagrid.org/mwiki/index.php?title=GTS:1.1:Administrators_Guide:SyncGTS:Configuration

Step 3: Installing GTS Trust Roots (Optional)

In order for SyncGTS to sync with a GTS service, it is required that the local environment trust the GTS service being synced with. In other words the local environment must trust the certificate authority that issued the GTS Service's credentials. Most distributions of SyncGTS are pre-configured to trust the GTS credentialing certificate authority for the grid in which the distribution is configured. If this is the case no further configuration is required. If this is not the case SyncGTS can easily be configured to trust other certificate authorities by placing a copy of the CA's certificate in the directory: *SYNC_GTS_LOCATION/ext/resources/certificates*. The CA certificate must be contained in PEM format and must be given a digit (0-9) extension. For example, to configure SyncGTS to trust a CA whose certificate is contained in the file cacert.pem, the file should be renamed to cacert.0 and copied to the directory: *SYNC_GTS_LOCATION/ext/resources/certificates*. In most cases you will also want to add an entry to the excluded CA list in SyncGTS's configuration file. For more details on the SyncGTS configuration file please consult the following website:

http://www.cagrid.org/mwiki/index.php?title=GTS:1.1:Administrators_Guide:SyncGTS:Configuration

Step 4: Deploying SyncGTS

Once you have configured a container (Globus or Tomcat) you need to deploy SyncGTS to that container. To deploy SyncGTS to a Globus container type the following from a command prompt:

```
%> cd SYNC_GTS_LOCATION
%> ant deployGlobus
```

To deploy SyncGTS to a secure Tomcat container type the following from a command prompt:

```
%> cd SYNC_GTS_LOCATION
%> ant deployGlobus
```

Regardless of which container you select, a significant amount of output is displayed to the screen. If the deployment is successful, *"BUILD SUCCESSFUL"* is displayed on the screen.

Grid Grouper

Grid Grouper provides a group based authorization solution for the grid, where grid services and applications enforce authorization policy based on membership to groups defined and managed at the grid level. Grid Grouper is built on top of Grouper, an internet initiative focused on providing tools for group management. Grouper is a java object model that currently supports: basic group management by distributed authorities; subgroups; composite groups (whose membership is determined by the union, intersection, or relative complement of two other groups); custom group types and custom attributes; trace back of indirect membership; and delegation. Applications interact with Grouper by embedding the Grouper's java object model within applications. Grouper does not provide a service interface for accessing groups. For more information on Grouper, refer to the following URL:

https://wiki.internet2.edu/confluence/display/GrouperWG/Home

Grid Grouper (Figure 6-34) is a grid enabled version of Grouper, providing a web service interface to the Grouper object model. Grid Grouper makes groups managed by Grouper available and manageable to applications and other services in the grid. Grid Grouper provides an almost identical object model to the Grouper object model on the grid client side. Applications and services can use the Grid Grouper object model much like they would use the Grouper object model to access and manage groups as well as enforce authorization policy based on membership to groups.

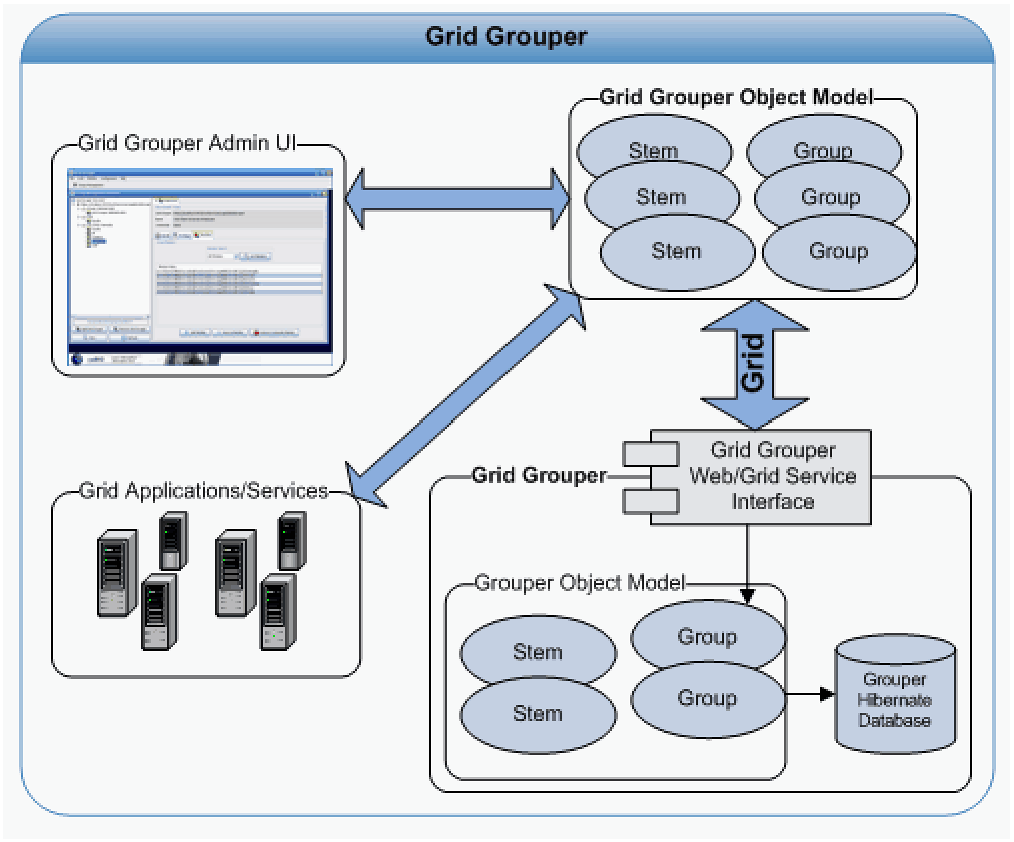

Grid Grouper
Grid Grouper Admin UI
Grid Grouper Object Model
Stem
Stem
Stem
Group
Group
Group
Grid
Grid Applications/Services
Grid Grouper
Grid Grouper Web/Grid Service Interface
Grouper Object Model
Stem
Stem
Group
Group
Grouper Hibernate Database

Figure 6-34 Grid Grouper Architecture

In Grouper/Grid Grouper, groups are organized into namespaces called stems. Each stem can have a set of child stems and a set of child groups with exception to the root stem, which cannot have any child groups. For example, consider a university that is comprised of many departments each of which has Faculty, Staff, and Students. In terms of organizing the university in Grid Grouper, a stem could be created for each department and each department stem would contain three groups; one each for Faculty, Staff, and Students.

Installation and Configuration

Grid Grouper is distributed as a standalone project as well as part of other projects such as caGrid. Each of the distributions contains a *gridgrouper* directory herein referred to as GRID_GROUPER_LOCATION. To install and configure Grid Grouper, use the following steps.

Step 1: Install Prerequisite Software

In order to install and run Grid Grouper, it is required that the prerequisite software in Table 6-3 is installed.

Software	**Version**	**Description**
Java SDK	jsdk1.5 or higher	Grid Grouper is written in Java therefore it requires the Java SDK. After installing you will have to set up an environmental variable pointing to the Java SDK

Software	Version	Description
		directory and name it JAVA_HOME.
MySQL	Mysql 4.1.x or higher	For persisting the trust fabric and other information.
Ant	Ant 1.6.5	GridGrouper along with the Globus Toolkit in which GridGrouper is built on, uses Jakarta Ant for building and deploying.
Globus	Globus 4.0.3	GridGrouper is built on top of the Globus Toolkit. GridGrouper requires the ws-core installation of the Globus Toolkit.
Tomcat (Only required if deploying to Tomcat)	Tomcat 5.0.28	GridGrouper can be optionally deployed as a Grid Service to a Tomcat deployed Globus Toolkit.

Table 6-3 Grid Grouper software prerequisites

Step 2: Building Grid Grouper

If you have obtained a source release of Grid Grouper you will need to build Grid Grouper. To build Grid Grouper type the following from a command prompt:

```
%> cd GRID_GROUPER_LOCATION
%> ant clean all
```

Note: Depending on the Grid Grouper distribution it may be required to build the entire project that Grid Grouper is distributed with prior to building Grid Grouper. For example if you have obtained a caGrid source distribution this is required. If you received a Grid Grouper standalone distribution this is not required.

Step 3: Obtain a Host Credential

Grid Grouper requires that it runs as a secure service. In order to run a secure service, the container hosting the service must run with a host credential. A host credential consists of a X.509 certificate and a private key. One of the features Dorian provides is the ability to issue and manage host credentials. There are many methods of retrieving host credentials, these methods include but are not limited to the following:

- Requesting a credential from a known/trusted certificate authority (caGrid Certificate Authority).
- Standing up a Dorian service.
- Standing up a simple certificate authority.

For production environments it is recommended that you obtain a host credential from a trusted certificate authority (option 1), such as a caGrid Certificate Authority. Standing up a Dorian (option 2) is another solid option especially if you wish to run your own production Certificate Authority. Standing up a simple certificate authority (option 3) is not recommended for production environments but is an excellent option for quickly obtaining a host credential for testing purposes.

If you have a host credential already or you have a method of obtaining (option 1 or option 2) one please proceed to the next step; otherwise for the purposes of this guide certificate authority (option 3) will be created and used to issue a host credential. To create a certificate authority, use the following steps from a command prompt (illustrated below):

cd GRID_GROUPER_LOCATION

Type *ant generateCA*

Enter the distinguished name (DN) for the CA (i.e O=xyz,OU=abc,CN=My CA).

Enter the number of days that the CA will be valid for (i.e 3650)

Enter a password which will be used to encrypt the CA's private key.

Enter a file to write the CA private key to.

Enter a file to write the CA certificate to.

```
%> cd GRID_GROUPER_LOCATION
%> ant generateCA
Buildfile: build.xml

setGlobus:

checkGlobus:
     [echo] Globus: C:\ext\ws-core-4.0.3

generateCA:
    [input] Please enter the DN for the new CA (ex.
O=xyz,OU=abc,CN=My CA):
O=xyz,OU=abc,CN=My CA
    [input] Please enter the number of days the new CA will be
valid for:
3650
    [input] Please enter a password for the new CA:
password
    [input] Please enter a location to write the new CA's private
key:
cakey.pem
    [input] Please enter a location to write the new CA's
certificate:
cacert.pem
     [java] Successfully create the CA certificate:
     [java] O=xyz,OU=abc,CN=My CA
     [java] CA Certificate Valid Till:
     [java] Fri Jun 23 12:47:10 EDT 2017
     [java] CA Private Key Written to:
     [java] cakey.pem
     [java] CA Certificate Written to:
     [java] cacert.pem

BUILD SUCCESSFUL
Total time: 46 seconds
```

Once a certificate authority is created, use it to issue a host credentials. To create host credentials, use the following steps from a command prompt (illustrated below):

cd GRID_GROUPER_LOCATION

Type *ant createAndSignHostCertificate*

Enter the location of the CA's private key.

Enter the password used to encrypt the CA's private key.

Enter the location of the CA's certificate.

Enter the name of the host.

Enter the number of days that the host credentials should be valid for.

Enter a location to write the host private key.

Enter a location to write the host certificate.

```
%> cd GRID_GROUPER_LOCATION
%> ant createAndSignHostCertificate
Buildfile: build.xml

setGlobus:

checkGlobus:
     [echo] Globus: C:\ext\ws-core-4.0.3

createAndSignHostCertificate:
    [input] Please enter the location of the CA's private key:
cakey.pem
    [input] Please enter the CA's password:
password
    [input] Please enter the location of the CA's certificate:
cacert.pem
    [input] Please enter the Hostname [${env.HOST}]:
myhost
    [input] Please enter the number of days the host certificate
will be valid f
or:
365
    [input] Please enter a location to write the host key:
hostkey.pem
    [input] Please enter a location to write the host certificate:
hostcert.pem
     [java] Successfully create the user certificate:
     [java] O=xyz,OU=abc,CN=host/myhost
     [java] User certificate issued by:
     [java] O=xyz,OU=abc,CN=My CA
     [java] User Certificate Valid Till:
     [java] Wed Jun 25 13:58:37 EDT 2008
     [java] User Private Key Written to:
     [java] hostkey.pem
     [java] User Certificate Written to:
     [java] hostcert.pem

BUILD SUCCESSFUL
Total time: 52 seconds
```

Step 4: Configure Globus To Trust Grid Grouper

In order to securely invoke Grid Grouper, configure Globus to trust the CA that issued the host credentials obtained in the previous step. Place a copy of the certificate for the CA that issued the host credentials in the Globus trusted certificates directory. Unless otherwise specified during installation, the Globus trusted certificate directory is usually *USER_HOME/.globus/certificates*. Globus requires all CA certificates in its trusted certificates directory to be in PEM format and to have a digit extension (0-9). For example, if a CA certificate is stored in the file *cacert.pem* in PEM format than in order to configure Globus to trust this certificate authority it should be copied in to the directory *USER_HOME/.globus/certificates* (create directory if needed) with the file name *cacert.0*

Step 5: Configuring a Secure Container

Once host credentials are obtained, use them to configure a secure container. Grid Grouper can be run from a secure Globus container or a secure Tomcat container. For directions on how to configure a secure Globus container please consult the following website:

http://www.cagrid.org/mwiki/index.php?title=CaGrid:How-To:SecureGlobusContainer

For directions on how to configure a secure Tomcat container please consult the following website:

http://www.cagrid.org/mwiki/index.php?title=CaGrid:ConfigureTomcat

Step 6: Configuring Grid Grouper

To configure GridGouper, specify your MySQL database information in the *grouper.hibernate.properties* configuration file located in *GRID_GROUPER_LOCATION/resources/conf/*. The properties you need to edit are highlighted in bold in below, mainly the database connection URL, database username, and database password.

```
#MySQL
hibernate.dialect
=net.sf.hibernate.dialect.MySQLDialect
hibernate.connection.driver_class     = com.mysql.jdbc.Driver
hibernate.connection.url              =
jdbc:mysql://localhost:3306/grouper
hibernate.connection.username         = root
hibernate.connection.password         = YOUR_PASSWORD
```

Once you have edited the Grid Grouper configuration file, initialize the Grid Grouper database by manually creating the grouper database in MySQL. Name the database as configured in the *hibernate.connection.url* property of the *grouper.hibernate.properties* configuration file. Once the database is created, enter *ant grouperInit* to build out and initialize the Grouper/Grid Grouper database.

Step 7: Adding Initial Grid Grouper Administrator(s)

In order to administrate Grid Grouper, Grid Grouper must be initially provided with at least one administrator. Grid Grouper provides a command line tool for bootstrapping it and initially adding administrator(s). To leverage this command line utility type the following from a command prompt:

```
%> cd GRID_GROUPER_LOCATION
%> ant addAmin
```

Step 8: Deploying Grid Grouper

Once you have configured a secure container (Globus or Tomcat), deploy Grid Grouper to that container by typing the following from a command prompt:

```
%> cd GRID_GROUPER_LOCATION
%> ant deployGlobus
```

To deploy Grid Grouper to a secure Tomcat container type the following from a command prompt:

```
%> cd GRID_GROUPER_LOCATION
%> ant deployTomcat
```

Regardless of which container you select, a significant amount of output is displayed. If the deployment is successful, *"BUILD SUCCESSFUL"* is displayed on the screen.

Step 9: Verifying the Installation

Once you have deployed Grid Grouper, the installation and configuration of Grid Grouper is complete. Before verifying that the installation was successful, start the Grid Grouper service by starting the container that Grid Grouper was deployed to. For directions on starting a secure Globus container please consult the following web site:

http://www.cagrid.org/mwiki/index.php?title=CaGrid:How-To:SecureGlobusContainer

To start a secure Tomcat container run the startup script (`startup.sh` or `startup.bat`) located in TOMCAT_INSTALLATION_DIRECTORY/bin. If the container starts, verify that the Grid Grouper installation was successful by typing the following from the command prompt:

```
%> cd GRID_GROUPER_LOCATION
%> ant ui
```

In the Grid Grouper Administration UI that opens, use the following steps:

1. Click **Group Browser** to open the Group Browser window (Figure 6-35).
2. Click **Add Grid Grouper** to open the Add Grid Grouper dialog box.
3. From the **Grid Grouper** drop down menu, select *https://localhost:8443/wsrf/services/cagrid/GridGrouper*
4. Click **Add**.

The UI adds the Grid Grouper, *https://localhost:8443/wsrf/services/cagrid/GridGrouper* to the *Grid Grouper Service(s)* tree and populates a stem/group hierarchy in a sub tree. In the hierarchy there should be one stem, *Grouper Administration,* and under that stem there should be one group, *Grid Grouper Administrators* (Figure 6-35). Grid Grouper has been installed and configured.

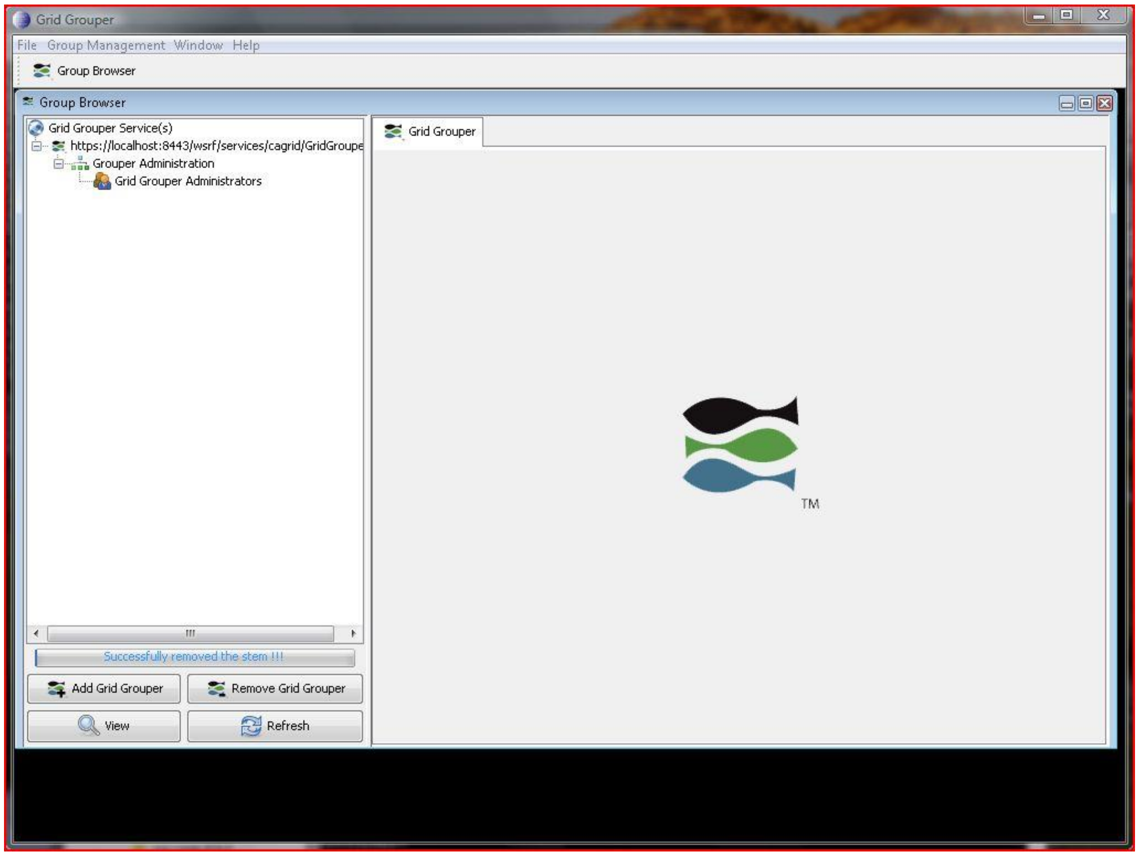

Grid Grouper
File Group Management Window Help
Group Browser
Group Browser
Grid Grouper Service(s)
https://localhost:8443/wsrf/services/cagrid/GridGroupe
Grouper Administration
Grid Grouper Administrators
Grid Grouper
Successfully removed the stem !!!
Add Grid Grouper
Remove Grid Grouper
View
Refresh
TM

Figure 6-35 Grid Grouper browser

Administrating Grid Grouper

Grid Grouper Administrators

Initially Grid Grouper has a root stem with one child stem named *Grouper Administration* (grouperadministration). The Grouper Administrative stem contains one group named *Grid Grouper Administrators* (grouperadministration:gridgrouperadministrators). The *Grid Grouper Administrators* group is the super user group for Grid Grouper; all members of this group have administrative privileges on all the stems and groups within Grid Grouper.

Note: The individual groups and stems can also be assigned administrators. The GAARDS UI provides a method of adding and removing administrators to/from the Grid Grouper Administrators group. The Grid Grouper Administrator's group can be managed like any other Grid Grouper group.

Administrating Stems

In Grouper/Grid Grouper, groups are organized into namespaces or stems. Each stem can have a set of child stems and a set of child groups with exception to the root stem which cannot have any child groups. The Stem hierarchy in Grid Grouper is publicly visible to anyone accessing the service; however, the ability to view a group within a stem publicly depends on the privileges for the group. A Stem can have two types of privileges associated with it: the "Stem Privilege" and the "Create Privilege". Users with the "Stem Privilege" can create, modify, and remove child stems. Users with the "Create Privilege" can create, modify, and remove child groups.

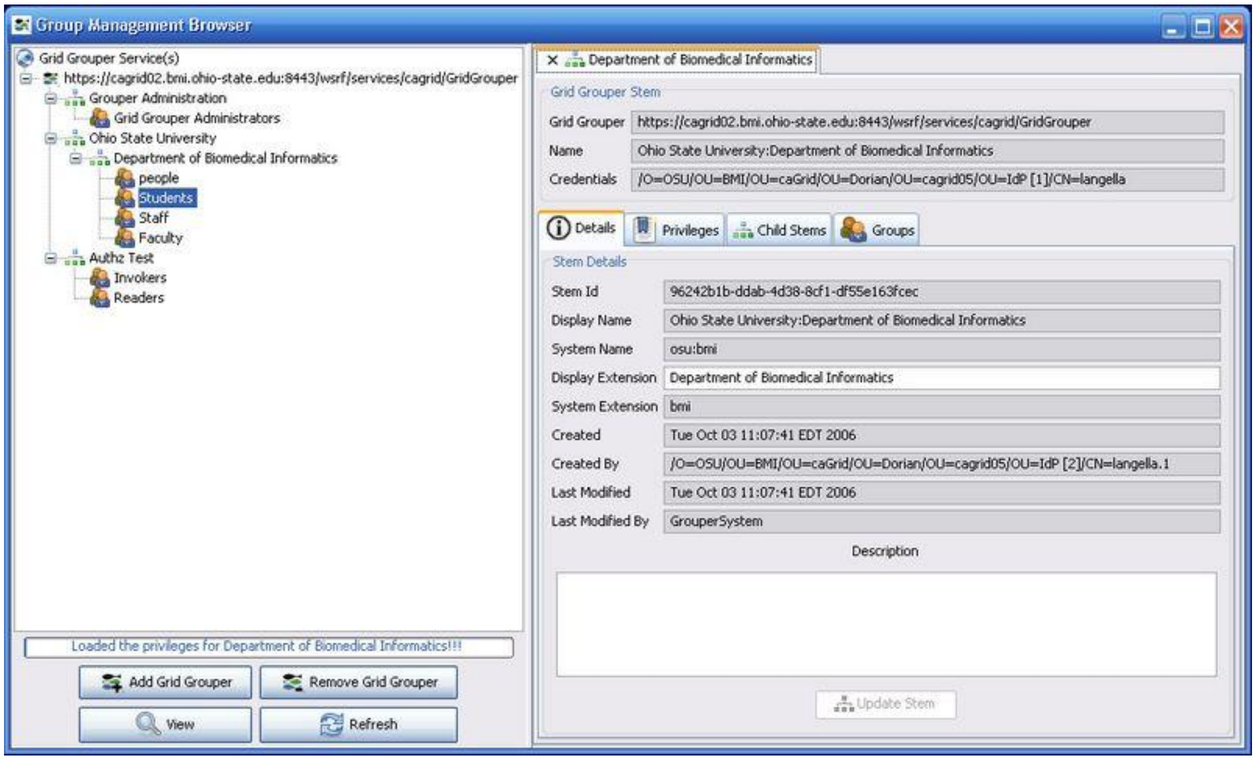
To administrate stems with the GAARDS UI, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a Grid Grouper or Stem Administrator).
3. Select **Group Management > Group Browser** to open the Group Browser window.
4. Click **Add Grid Grouper** to open the Add Grid Grouper dialog box.
5. From the **Grid Grouper** drop down menu, select the URI of the Grid Grouper you wish to administrate.
6. From the **Credentials** drop down menu, select the credential you wish to use to administrate.
7. Click **Add** to load the Grid Grouper you specified into the Group Browser window.
8. Select the stem you want to administer and click **View**.

A tab opens entitled with the stem's name in the Details pane (Figure 6-35). This tab will be referred to as the Stem Administration tab. The top of this tab lists the Grid Grouper in which the stem exists, the full display name of the stem in regards to the rest of the hierarchy, and the credentials you used to obtain the stem. It also contains four sub tabs: Details, Privileges, Child Stems, and Groups. The Details tab lists a stem's metadata which includes:

- Stem Id – Unique Id assigned to the Stem by Grouper.
- Display Name – Full display name for the stem with context to the rest of the hierarchy.
- System Name – Full system name for the stem with context to the rest of the hierarchy.
- Display Extension – Local display name for the stem.
- System Extension – Local system name for the stem.
- Create – Date the stem was created.
- Create By – The identity of the user or service that created the stem.
- Last Modified – Date the stem was last modified.
- Last Modified By– The identity of the user or service who last modified the stem.
- Description – Human readable description of the stem.

Of the metadata listed, only the display extension and description may be updated. To update the metadata make the necessary changes and click **Update Stem**.



Loaded the privileges for Department of Biomedical Informatics!!!

Grid Grouper	https://cagrid02.bmi.ohio-state.edu:8443/wsrf/services/cagrid/GridGrouper
Name	Ohio State University:Department of Biomedical Informatics
Credentials	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella

Stem Id	96242b1b-ddab-4d38-8cf1-df55e163fcec
Display Name	Ohio State University:Department of Biomedical Informatics
System Name	osu:bmi
Display Extension	Department of Biomedical Informatics
System Extension	bmi
Created	Tue Oct 03 11:07:41 EDT 2006
Created By	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1
Last Modified	Tue Oct 03 11:07:41 EDT 2006
Last Modified By	GrouperSystem

Figure 6-36 Group Management Browser

Stem Privileges

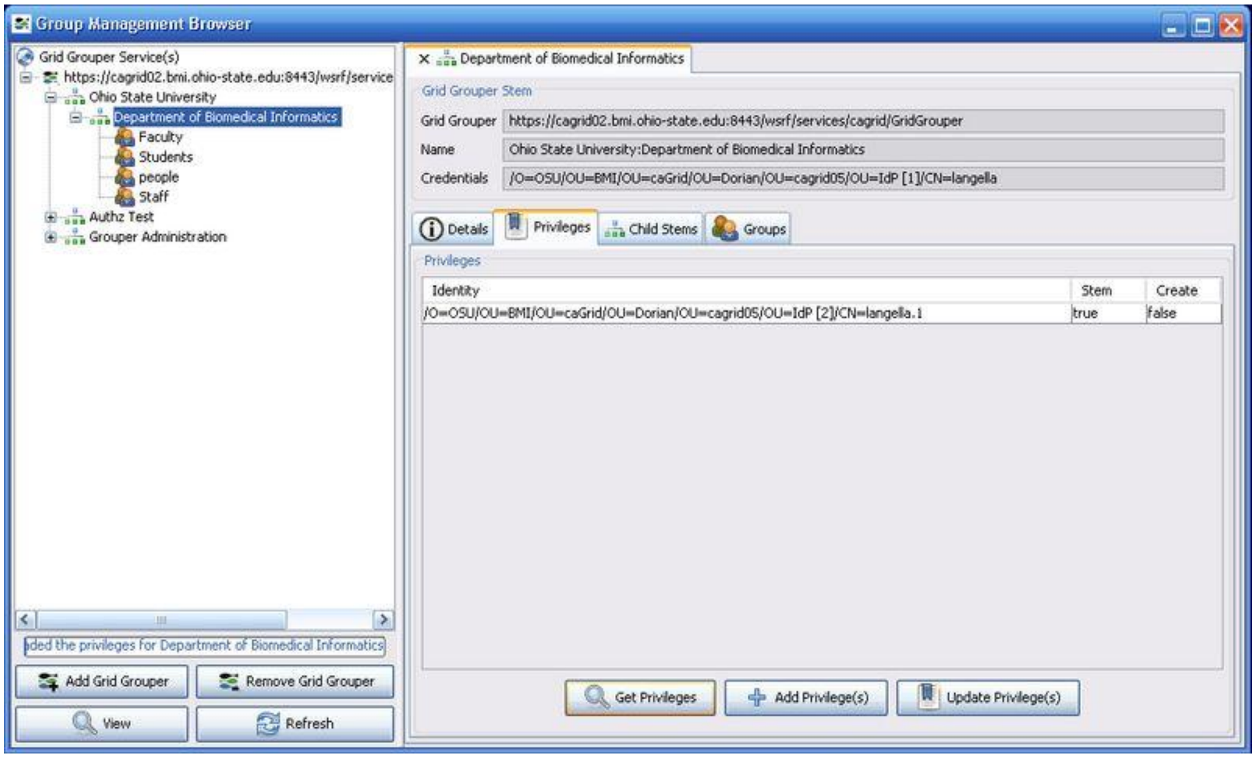
The stem hierarchy in Grid Grouper is publicly visible to anyone accessing the service; however the ability to view a group within a stem depends on the privileges for the group. A stem can have two types of privileges associated with it: the *"Stem Privilege"* and the *"Create Privilege"*. Users with the *"Stem Privilege"* can create, modify, and remove child stems. Users with the *"Create Privilege"* can create, modify, and remove child groups.

The GAARDS UI provides the ability to administrate stem privileges. To list all the existing privileges granted on a stem use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a Grid Grouper or Stem Administrator).
3. Select **Group Management > Group Browser** to open the Group Browser window.
4. Click **Add Grid Grouper** to open the Add Grid Grouper dialog box.
5. From the **Grid Grouper** drop down menu, select the URI of the Grid Grouper you wish to administrate. From the **Credentials** drop down menu, select the credential you wish to use to administrate.
6. Click **Add** to load the Grid Grouper you specified into the Group Browser window (Figure 6-37). Select the stem to administer and click **View**.

7. Select the **Privileges** tab and click **Get Privileges**.

All the users with privileges on the stem and the privileges that each user has are listed. For example in Figure 6-37, the stem shown lists one user that has been assigned privilege(s); the user listed has been assigned the Stem privilege.



Grid Grouper Stem

Grid Grouper: https://cagrid02.bmi.ohio-state.edu:8443/wsrf/services/cagrid/GridGrouper

Name: Ohio State University:Department of Biomedical Informatics

Credentials: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella

Privileges

Identity	Stem	Create
/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1	true	false

dded the privileges for Department of Biomedical Informatics

Figure 6-37 Listing of stem privileges

Adding/Revoking Privileges

Users without any existing privileges and users with existing privileges can be granted/revoked privileges by using the following steps:

1. For user's without any existing privileges, click **Add Privilege(s)**. For users with existing privileges, select the user from the privileges table and click **Update Privilege**. The Update Stem Privilege(s) window opens (Figure 6-38).
2. If you are granting privileges to a user without existing privileges, specify the user's grid identity in the **Grid Identity** text field.
3. To grant privileges, select the privileges you wish to grant. To revoke privileges, deselect the privileges you wish to revoke.
4. Click **Update Privilege(s)**.

Changes to Grid Grouper are committed and are effective immediately.

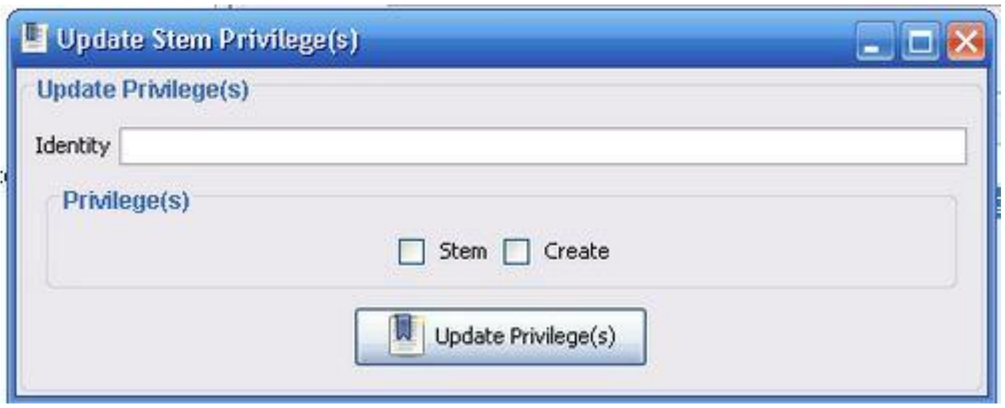
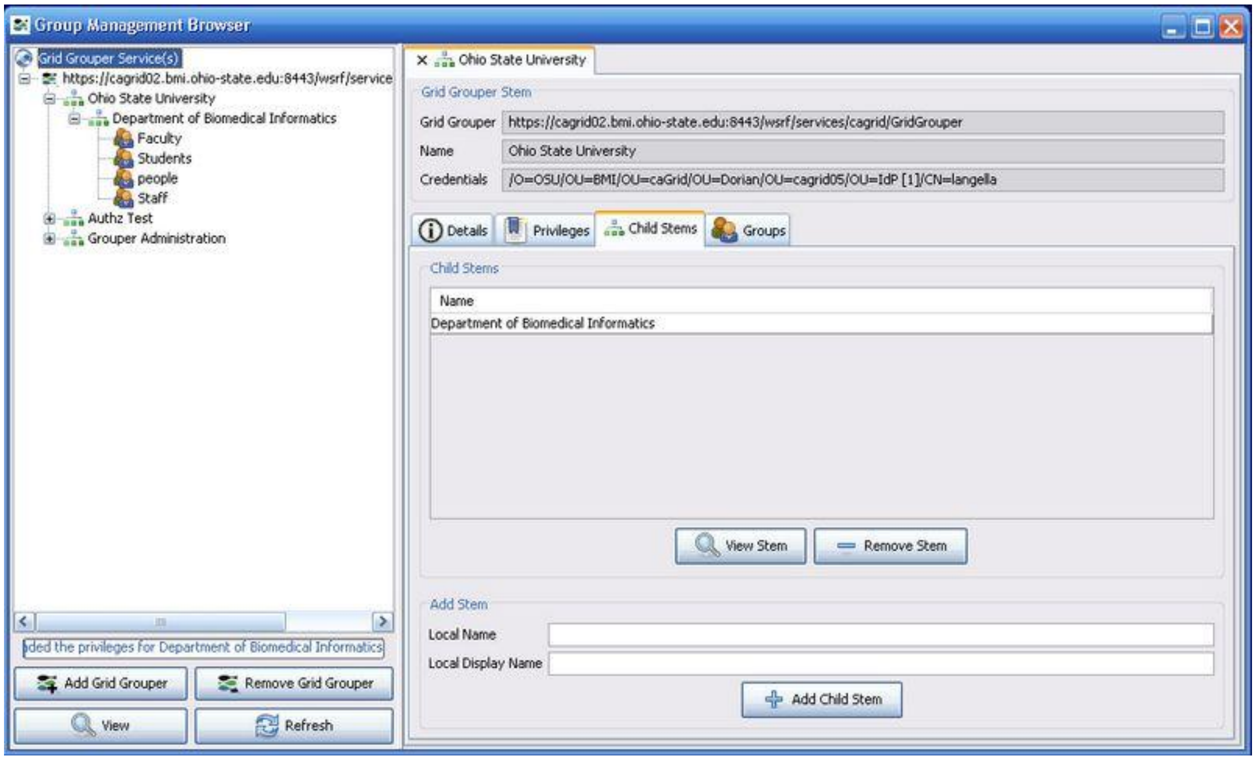


Figure 6-38 Update Stem Privileges window

Managing Child Stems

Each stem in Grid Grouper can have a set of child stems. The GAARDS UI provides a means of listing, creating, and removing child stems. Since stems are publicly readable any user may view the stem hierarchy; however only users with the *"Stem Privilege"* may create and remove stems. To view the child stems for a given stem, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a Grid Grouper or Stem Administrator).
3. Select **Group Management > Group Browser** to open the Group Browser window.
4. Click **Add Grid Grouper** to open the Add Grid Grouper dialog box (Figure 6-39).
5. From the **Grid Grouper** drop down menu, select the URI of the Grid Grouper you wish to administrate.
6. From the **Credentials** drop down menu, select the credential you wish to use to administrate.
7. Click **Add** to load the Grid Grouper you specified into the Group Browser window.
8. Select the stem you want to administer and click **View** to open the **Stem Administration** tab for the selected stem.
9. Select the **Child Stems** tab; the stem's child stems is listed in the **Child Stems** table.



dded the privileges for Department of Biomedical Informatics

Grid Grouper Stem

Grid Grouper	https://cagrid02.bmi.ohio-state.edu:8443/wsrf/services/cagrid/GridGrouper
Name	Ohio State University
Credentials	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella

Child Stems

Name
Department of Biomedical Informatics

Add Stem

Local Name	
Local Display Name	

Figure 6-39 Managing child stems in the Group Management Browser

Viewing a Child Stem

To view a child stem, use the following steps:

1. Select the stem to view from the **Child Stems** table.
2. Click **View Stem** to open the Stem Administration tab for the selected stem.

Adding a Child Stem

To add a child stem, use the following steps:

1. Enter a Local Name for the stem in the **Local Name** text field.
2. Enter a Local Display Name for the stem in the **Local Display Name** text field.
3. Click **Add Child Stem**.

Removing a Child Stem

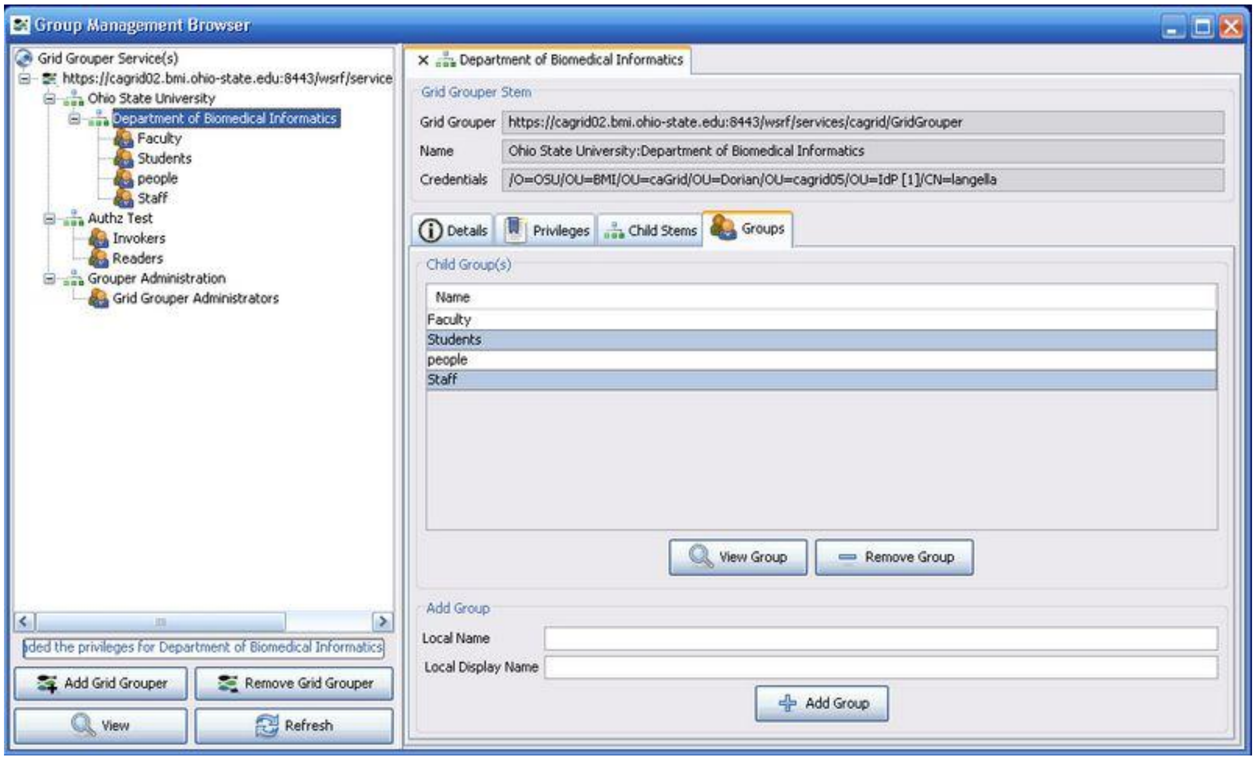
To remove a child stem, use the following steps:

1. Select the stem to remove from the **Child Stems** table.
2. Click **Remove Stem.**

Managing Groups

Each stem in Grid Grouper can have a set of groups. The GAARDS UI provides a means of listing, creating, and removing groups; however only users with the *"Create Privilege"* may create and remove groups. To view the child groups for a given stem, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a Grid Grouper or Stem Administrator).
3. Select **Group Management > Group Browser** to open the Group Browser window.
4. Click **Add Grid Grouper** to open the Add Grid Grouper dialog box.
5. From the **Grid Grouper** drop down menu, select the URI of the Grid Grouper you wish to administrate.
6. From the **Credentials** drop down menu, select the credential you wish to use to administrate.
7. Click **Add** to load the Grid Grouper you specified in the Group Browser window.
8. Select the stem you want to administer and click **View** to open the Stem Administration tab for the selected stem.
9. Select the **Groups** tab; the stem's groups are listed in the Child Group(s) table.


Group Management Browser

Grid Grouper Service(s)
- https://cagrid02.bmi.ohio-state.edu:8443/wsrf/service
  - Ohio State University
    - Department of Biomedical Informatics
      - Faculty
      - Students
      - people
      - Staff
  - Authz Test
    - Invokers
    - Readers
  - Grouper Administration
    - Grid Grouper Administrators

ded the privileges for Department of Biomedical Informatics

Department of Biomedical Informatics

Grid Grouper Stem

Grid Grouper: https://cagrid02.bmi.ohio-state.edu:8443/wsrf/services/cagrid/GridGrouper

Name: Ohio State University:Department of Biomedical Informatics

Credentials: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella

Child Group(s)

Name
Faculty
Students
people
Staff

Add Group

Local Name:

Local Display Name:

Figure 6-40 Managing groups in the Group Management Browser

Viewing a Group

To view a group, use the following steps:

1. Select the group to view from the Child Group(s) table.
2. Click **View Group** to open the Group Administration tab for the selected group.

Adding a Group

To add a group, use the following steps:

1. Enter a Local Name for the stem in the **Local Name** text field.
2. Enter a Local Display Name for the stem in the **Local Display Name** text field.
3. Click **Add Group**.

Removing a Group

To remove a group, use the following steps:

1. Select the group to remove from the Child Group(s) table.
2. Click **Remove Group**.

Administrating Groups

In Grouper/Grid Grouper groups are comprised of a set of metadata describing the group, a set of members in the groups, and a set of privileges assigned to users for protecting access to the group. Grid Grouper provides three mechanisms for adding members to a group: 1) directly adding a member 2) adding a subgroup to a group 3) making a group a composite of other groups. Directly adding a user as a member to a group is straight forward; these members are referred to as "Immediate Members". Adding a subgroup to a group makes all the members of the subgroup members of the group in which the subgroup was added. Members in a group whose membership is granted by membership in a subgroup are referred to as "Effective Members". A group can also be set to a "Composite" group. A composite group consists of a set operation (Union, Intersection, Complement) on two other groups. For example, a composite group consisting of the Intersection of Group X and Group Y would contain all the members that are both members of Group X and Group Y. Members whose membership is granted through a composite group are referred to as "Composite Members".

To protect access to groups in Grid Grouper, users can be assigned the following privileges on a group: View, Read, Update, Admin, Optin, and Optout. Users with the View privilege can see that the group exists. Users with the Read privilege can read basic information about the group. Users with the Update Privilege can manage memberships to the group as well as administer View, Read, and Update privileges. Users with the Admin privilege can modify/administer anything on the group: metadata, privileges, and memberships. Users with the Optin privilege can add themselves as a member to a group, similarly users with the Opout privilege can remove themselves from a group. By default Grid Grouper grants Read and View privileges to all users on each group.

To administrate groups with the GAARDS UI complete the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a Grid Grouper or Stem

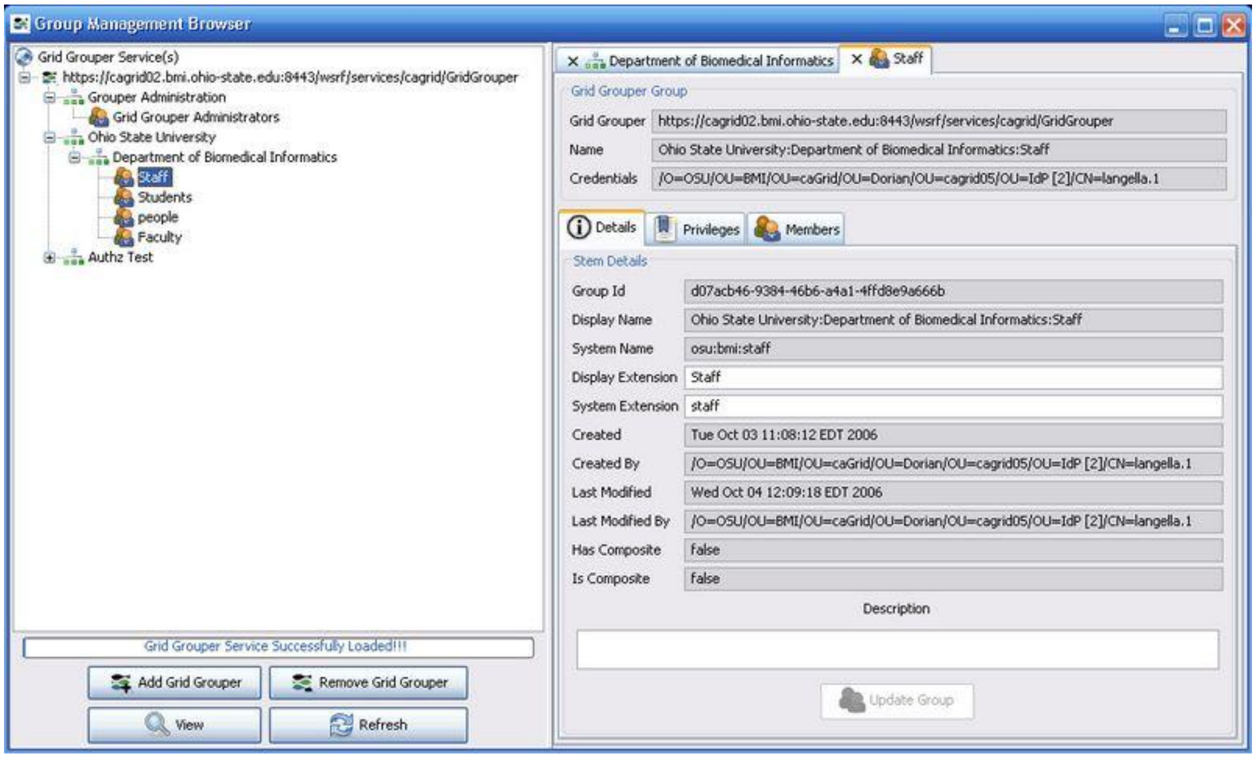
Administrator).

3. Select **Group Management > Group Browser** to open the Group Browser window.
4. Click **Add Grid Grouper** to open the Add Grid Grouper dialog box.
5. From the **Grid Grouper** drop down menu, select the URI of the Grid Grouper you wish to administrate.
6. From the **Credentials** drop down menu, select the credential you wish to use to administrate.
7. Click **Add** to load the Grid Grouper you specified into the Group Browser window.
8. Select the group you want to administer and click **View**.

A tab, entitled with the group's name in the details pane opens. This tab is referred to as the *Group Administration* tab. The top of the tab lists the Grid Grouper in which the group exists, the full display name of the group in regards to the rest of the hierarchy, and the credentials used to obtain the group. The tab also contains three sub tabs: *Details*, *Privileges*, and *Members*. The *Details* tab lists a group's metadata, which includes:

- Group Id – Unique Id assigned to the group by Grouper.
- Display Name – Full display name for the group with context to the rest of the hierarchy.
- System Name – Full system name for the group with context to the rest of the hierarchy.
- Display Extension– Local display name for the group.
- System Extension– Local system name for the group.
- Create– Date the group was created.
- Create By– The identity of the user or service that created the group.
- Last Modified– Date the group was last modified.
- Last Modified By– The identity of the user or service who last modified the group.
- Description – Human readable description of the group.

Of the metadata listed, only the display extension, system extension, and description can be updated by making the changes, and clicking the **Update Group** button.



Grid Grouper Service Successfully Loaded!!!

Grid Grouper Group

Grid Grouper	https://cagrid02.bmi.ohio-state.edu:8443/wsrf/services/cagrid/GridGrouper
Name	Ohio State University:Department of Biomedical Informatics:Staff
Credentials	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1

Stem Details

Group Id	d07acb46-9384-46b6-a4a1-4ffd8e9a666b
Display Name	Ohio State University:Department of Biomedical Informatics:Staff
System Name	osu:bmi:staff
Display Extension	Staff
System Extension	staff
Created	Tue Oct 03 11:08:12 EDT 2006
Created By	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1
Last Modified	Wed Oct 04 12:09:18 EDT 2006
Last Modified By	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1
Has Composite	false
Is Composite	false

Description

Figure 6-41 Administrating groups

Group Privileges

To protect access to groups in Grid Grouper, users can be assigned the following privileges on a group:

- ***View*** - Allows user's to see that the group exists.
- ***Read*** - Allows user's to read basic information about the group.
- ***Update*** - Allows user's to manage memberships to the group as well as administer *View*, *Read*, and *Update* privileges.
- ***Admin*** - Allows modification/administration of all aspects of the Group including: metadata, privileges, and memberships.
- ***Optin*** - Allows user's to add themselves to a group.
- ***Optout*** - Allows user's to remove themselves from a group.

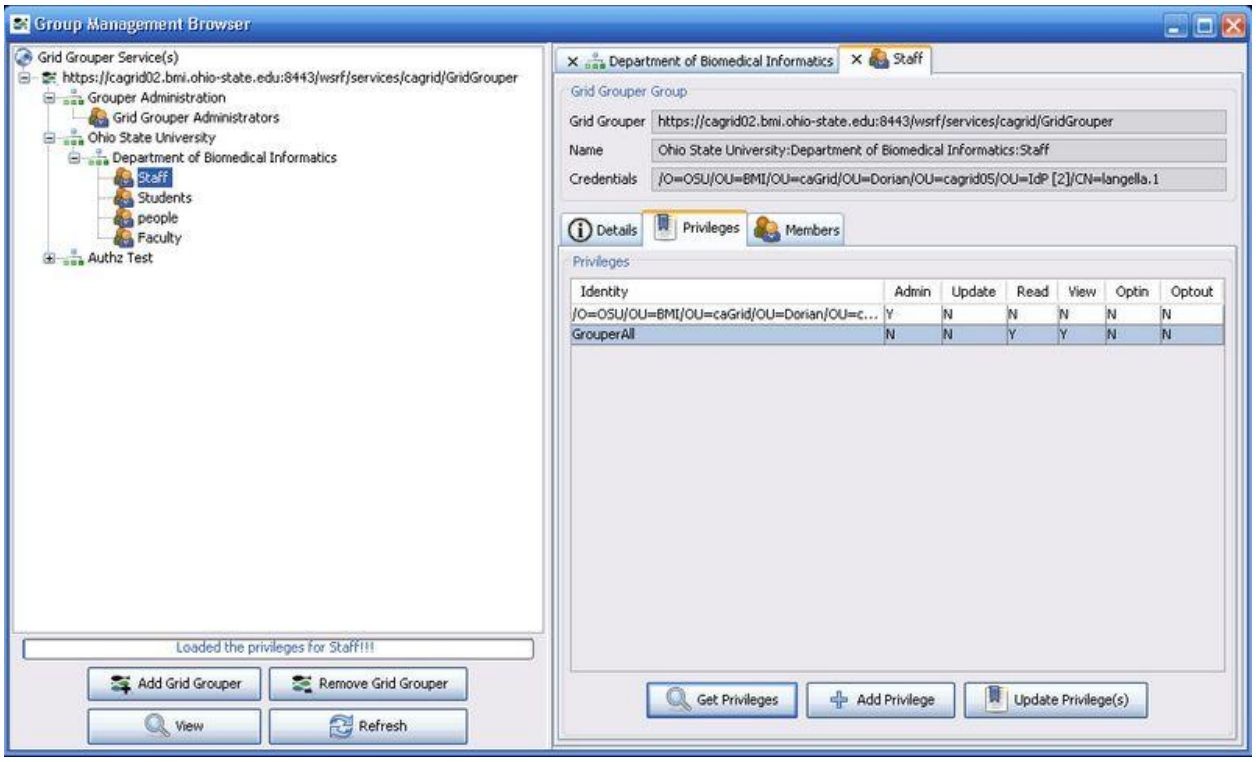
When a user accesses a group they will only be allowed to access the privileges assigned to them. Users without any privileges assigned inherit the privileges assigned to the *GrouperAll* user or default user. By default the *GrouperAll* is granted *Read* and *View* privileges on each group.

To list all the existing privileges granted on a group, use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account.

3. Select **Group Management > Group Browser** to open the Group Browser window.
4. Click **Add Grid Grouper** to open the Add Grid Grouper dialog.
5. From the **Grid Grouper** drop down menu, select the URI of the Grid Grouper you wish to administrate. From the **Credentials** drop down menu, select the credential you wish to use to administrate.
6. Click **Add** to load the Grid Grouper you specified into the Group Browser window.
7. Select the group you want to administer and click **View**.
8. Select the **Privileges** tab and click **Get Privileges**.

All the users with privileges on the group and the privileges that each user has are listed. For example, in Figure 6-42, the group shown lists two users that have been assigned privilege(s); the first user has been assigned the *Admin* privilege and the second user, *GrouperAll,* has been assigned *Read* and *Write* privileges.



Grid Grouper Group

Grid Grouper: https://cagrid02.bmi.ohio-state.edu:8443/wsrf/services/cagrid/GridGrouper
Name: Ohio State University:Department of Biomedical Informatics:Staff
Credentials: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1

Privileges

Identity	Admin	Update	Read	View	Optin	Optout
/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=c...	Y	N	N	N	N	N
GrouperAll	N	N	Y	Y	N	N

Figure 6-42 Managing group privileges

Adding/Revoking Privileges

Users without any existing privileges and users with existing privileges can be granted/revoked privileges by using the following steps:

1. For user's without any existing privileges, click **Add Privilege(s)**. For users with existing privileges, select the user from the Privileges table and click **Update Privilege(s)**. The

Update Group Privilege(s) window.

2. If you are granting privileges to a user without existing privileges, specify the user's grid identity in the **Grid Identity** text field.
3. To grant privileges select the privileges you wish to grant. To revoke privileges unselect the privileges you wish to revoke.
4. Click **Update Privilege(s)**.

Changes to Grid Grouper are committed and are effective immediately.

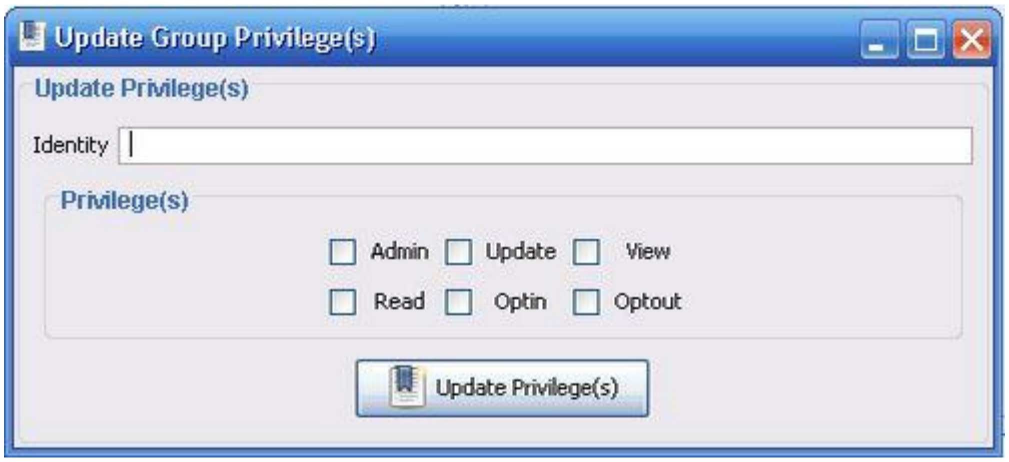


Figure 6-43 Update Group Privileges window

Group Memberships

Grid Grouper supports three types of group memberships:

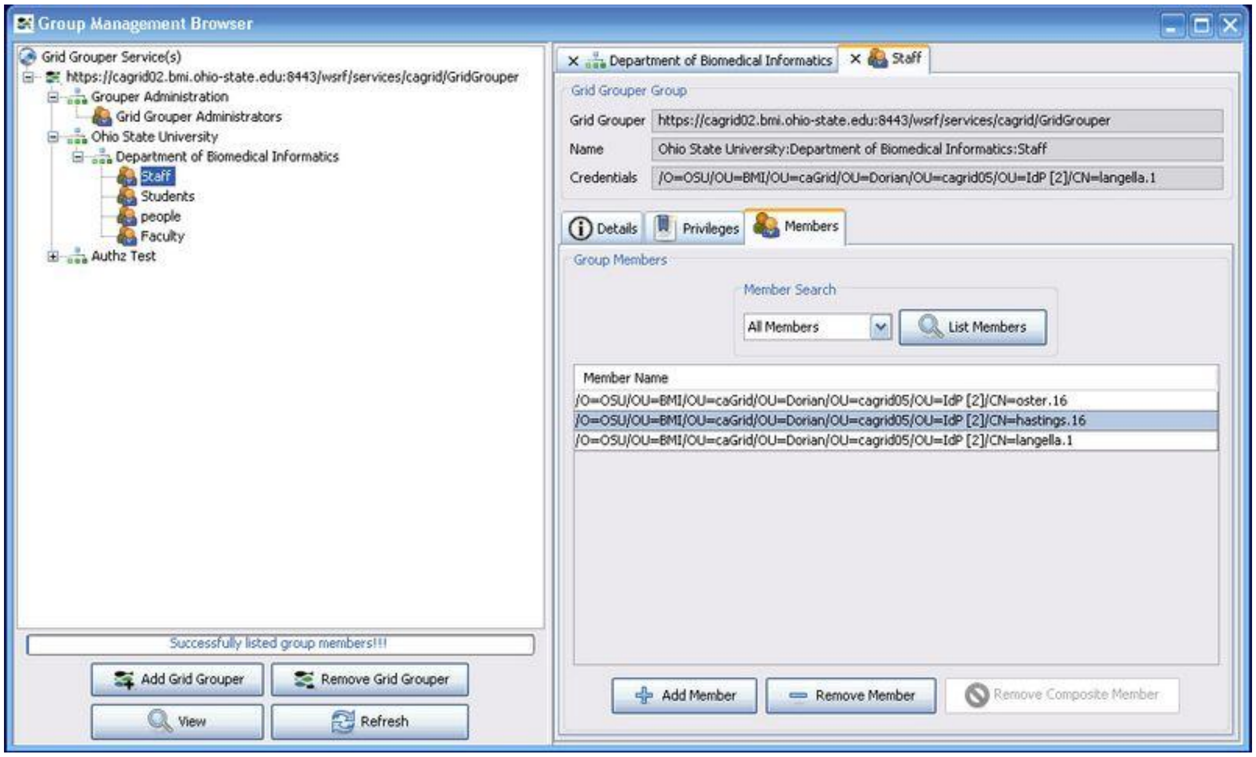
- Immediate Membership - Directly adding a member to a group.
- Effective Membership - Adding an existing group to a group as a subgroup. Adding a subgroup to a group makes all the members of the subgroup members of the group in which the subgroup was added. Members in a group whose membership is granted by membership in a sub group are referred to as *Effective Members*.
- Composite Membership - Membership is based on a set operation (Union, Intersection, or Complement) on two other groups. For example a composite group consisting of the Intersection of Group X and Group Y would contain all the members that are both members of Group X and Group Y. Members whose membership is granted through a composite group are referred to as *Composite Members*.

The GAARDS UI provides a means of listing, adding, and removing members from groups. To view the members of a given group use the following steps:

1. Launch the GAARDS UI.
2. Logon to the Grid using your user account (You must be a Grid Grouper or Stem Administrator).
3. Select **Group Management > Group Browser** to open the Group Browser window.

4. Click **Add Grid Grouper** to open the Add Grid Grouper dialog box.
5. From the **Grid Grouper** drop down menu select the URI of the Grid Grouper you wish to administrate. From the **Credentials** drop down menu select the credential you wish to use to administrate.
6. Click **Add** to load the Grid Grouper you specified into the Group Browser window.
7. Select the group you want to administer and click **View**.
8. Select the **Members** tab. Click **List Members**.

All the members of the group are listed. A member search can list all the members of the group or can list the members of the group by membership type (Immediate, Effective, Composite).



Grid Grouper: https://cagrid02.bmi.ohio-state.edu:8443/wsrf/services/cagrid/GridGrouper
Name: Ohio State University:Department of Biomedical Informatics:Staff
Credentials: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1

Member Search: All Members

Member Name
/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=oster.16
/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=hastings.16
/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1

Figure 6-44 Managing group memberships

Adding a Member

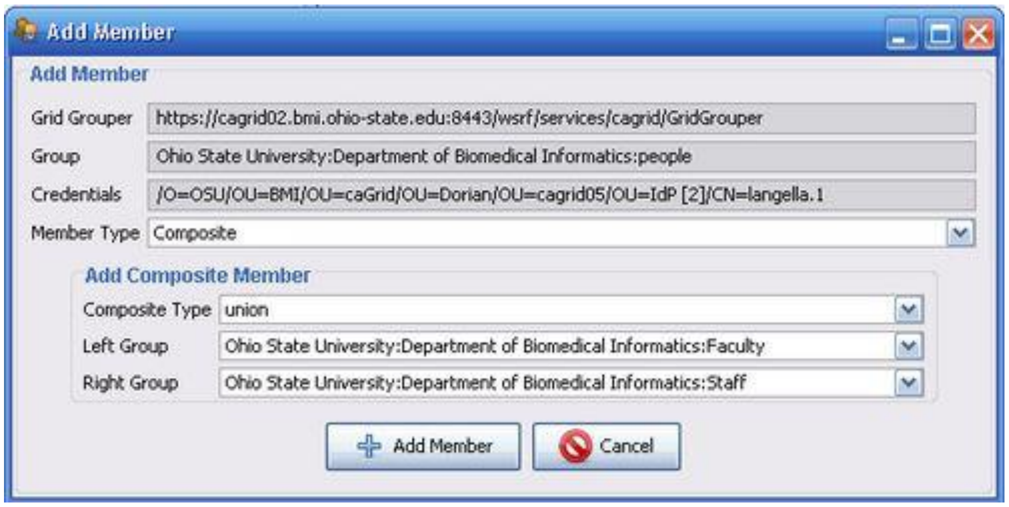
To add a member to a group use the following steps:

1. Click **Add Member** to open the Add Member window.
2. From the **Member Type**, select whether to add a **User**, **Group**, or **Composite** as a member.
3. If you chose User, enter the grid identity of that user in the **Member Identity** text field. If you chose Group, select the group from the **Group** drop down menu. If you chose Composite, 1) select the composite type from the **Composite Type** drop down menu, 2)

select a group from the **Left Group** drop down menu, and 3) select a group from the **Right Group** drop down menu.

4. Click **Add Member**.

In Figure 6-45, the user is adding a composite member consisting of the union of the staff group and the faculty group. It is important to note that a group with a composite membership, also referred to as a composite group, may only have one membership, which is the defined composite. A composite group may not contain additional immediate, effective, or composite members.



Grid Grouper: https://cagrid02.bmi.ohio-state.edu:8443/wsrf/services/cagrid/GridGrouper
Group: Ohio State University:Department of Biomedical Informatics:people
Credentials: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1
Member Type: Composite

Add Composite Member
Composite Type: union
Left Group: Ohio State University:Department of Biomedical Informatics:Faculty
Right Group: Ohio State University:Department of Biomedical Informatics:Staff

Figure 6-45 Adding a member to a group

Removing a Member

To remove a member from a group, use the following steps:

1. Select the member to remove from the **Members** table.
2. Click **Remove Member**.

Members whose membership to a group is obtained through being a member of a subgroup (Effective Membership) whose membership is obtained through a composite cannot be directly removed using the method just described. To remove effective members of a group, the member must be removed from the subgroup that they are immediate members of. To remove composite members from a group, the composite membership associated with the group must be removed by clicking the **Remove Composite Member** button.

Authentication Management

The role of the Authentication Service project is to provide an integration point between local identity management and caGrid identify federation. For example, your institution already manages local identities (user accounts) in an LDAP server, or RDBMS, or some other identity management system. In order for a user at your institution to obtain grid credentials (so that he can authenticate to secure caGrid services), the user must present a caGrid Dorian service with a SAML authentication assertion, indicating that the user has successfully authenticated to your institution's identity management system. That assertion must be digitally signed by an identity that the Dorian service trusts.

Information about the policy process for adding your institution's asserting credentials to a caGrid Dorian service can be found here: http://gforge.nci.nih.gov/projects/swg/. Information about the technical process of adding a trusted identity provider to a Dorian service using the GAARDS security user interface can be found here: http://www.cagrid.org/mwiki/index.php?title=Dorian:1.1:Administrators_Guide:Managing_Trusted_Identity_Providers

The AuthenticationService project defines a framework that can be used to create a service which will interact with your institution's local identity management system and create SAML assertions that can be presented to a caGrid Dorian service. Creating such a service enables your institution's Identity Provider (IdP) to be "plugged-in" to Dorian, which is the caGrid Identity Federation Service (IFS).

The AuthenticationService project provides a default implementation of that framework. This implementation uses the NCICB's Common Security Module (CSM). Out of the box, CSM can be easily configured to work with most LDAP or RDBMS identity management systems. (AuthenticationService requires version 3.2.1 of CSM).

The following sections describe how to configure and deploy the default CSM AuthenticationService implementation. Details about configuring CSM can be found here: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/csm. For information about creating a custom implementation of the AuthenticationService framework, see the caGrid 1.1 Programmer's Guide.

Configuring the Service

The AuthenticationService needs to know the CSM *application context name* in order to determine how to validate a user's credentials (i.e. username and password). It also needs information about the X.509 certificate and key (the asserting credentials) it should use to sign SAML assertions. This information should be specified in a Java properties file named *idp.properties* found in the root directory of the AuthenticationService project (i.e. caGrid/projects/authentication-service). This directory will be referred to simply as *SRC*.

Table 6-4 indicates what properties in this file must be edited.

Property	Description
csm.app.context	The name of the application context that contains the CSM authentication policy. This value must map to an application name specified in the JAAS configuration file (described later).
saml.provider.crt	The absolute path to the X.509 certificate that the Authentication Service should use to sign SAML assertions. This file must be in PEM format.
saml.provider.key	The absolute path to the X.509 private key.

Property	Description
saml.provider.pwd	The password for the private key. You need this only if the key is encrypted. (If there is no password, this value is ignored.)

Table 6-4 Properties in SRC/idp.properties

Deploying to the Container

The Authentication Service may be deployed to either a Globus standalone container or the Globus web application (deployed in Tomcat 5.0.28). It must be deployed to a secure container.

The *deployGlobus* Ant target, defined in `SRC/buid-deploy.xml`, deploys the service to the standalone Globus container pointed to by the GLOBUS_LOCATION environment variable. The *deployTomcat* Ant target deploys the service to the Globus web application in the Tomcat installation pointed to by the CATALINA_HOME environment variable.

Configuring the CSM

The default Authentication Service implementation uses CSM to validate a user's credentials and retrieve certain user attributes that are required by the caGrid Dorian service. CSM uses the Java Authentication and Authorization Service (JAAS) to enable modules that are responsible for authentication to be plugged in using a standardized approach. This section provides examples of how to configure CSM's JAAS *login modules* for both LDAP and RDBMS.

Note: JAAS provides a flexible configuration mechanism. This section describes only one approach (in particular, it describes the approach used by the caGrid installer.) For details on configuring JAAS, see:
http://java.sun.com/j2se/1.5.0/docs/guide/security/jaas/JAASRefGuide.html

The SAML authentication assertion that must be presented to Dorian in order to retrieve grid credentials must contain information about the user in the form of SAML attributes. These attributes correspond to the following information:

- Uid: the user's account name at his institution
- First Name
- Last Name
- Email Address

Thus, CSM must be configured to retrieve that information from either the LDAP server or RDBMS. The JAAS configuration file that configures CSM in this way should be named `.java.login.config` and placed in the home directory of the user account that the Tomcat or the Globus container is running under. This file must contain an entry with a JAAS application name that maps to the CSM application context name that was specified as the value of the `csm.app.context` property in `SRC/idp.properties`.

Figure 6-46 shows an example JAAS configuration file that configures the CSM RDBMSLoginModule.

```
myapp{
gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule
required
   driver="org.gjt.mm.mysql.Driver"
   url="jdbc:mysql://somehost:3306/somedatabase"
   user="dbuser"
   passwd="dbpassword"
   TABLE_NAME="CSM_USER"
  USER_LOGIN_ID="LOGIN_NAME"
  USER_PASSWORD="PASSWORD"
  USER_FIRST_NAME="FIRST_NAME"
  USER_LAST_NAME="LAST_NAME"
  USER_EMAIL_ID="EMAIL_ID"
  encryption-enabled=”YES”;
};
```

Figure 6-46 Example JAAS configuration file for configuring CSM RDBMSLoginModule

In the above configuration, the JAAS application name is *myapp*. The *driver*, *url*, *user*, and *passwd* parameters configure the JDBC connection to the RDBMS. In this case, a MySQL driver is being used. The appropriate JDBC driver for your RDBMS should be placed in `SRC/lib` before deploying the service.

The *TABLE_NAME*, *USER_LOGIN_ID*, *USER_PASSWORD*, *USER_FIRST_NAME*, *USER_LAST_NAME*, and *USER_EMAIL_ID* parameters indicate how the user’s credentials can be validated and the appropriate attributes retrieved.

If CSM is being used to manage these accounts, then one can configure if encryption should be used when validating the user’s password.

Figure 6-47 shows an example JAAS configuration file that configures the CSM LDAPLoginModule.

```
myapp{
    gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule required
    ldapHost="ldaps://my.ldap.host.org:636"
    ldapSearchableBase="ou=some,o=base"
    ldapUserIdLabel="cn"
    USER_FIRST_NAME="givenName"
    USER_LAST_NAME="sn"

    USER_EMAIL_ID="mail";

};
```

Figure 6-47 Example JAAS configuration file for configuring CSM LDAPLoginModule

Once the JAAS configuration file has been created and placed in the user's home directory, the Globus or Tomcat container can be started.

Authorization Management

The caGrid Authorization (Authz) component provides an integration point between local authorization policy and grid-wide authorization policy. Authorization policy in caGrid can be based on membership in groups that are defined in Grid Grouper. Authorization policy within an organization is usually based on an individual's identity within that organization. The Authz component provides a framework to map groups that have been defined in the NCICB's Common Security Module (CSM) with groups that have been defined in a Grid Grouper service. The result is that local CSM administrators can extend access privileges to members of the caBIG community based on membership in Grid Grouper groups, rather than having to create local identities for each individual.

Since the Authz component has been designed to plug into the CSM framework, caCORE systems that use CSM can plug in the Authz component without changing code. (Authz requires version 3.2.1 of CSM).

This section describes how to configure CSM for a caCORE service to use the Authz component. A detailed, step-by-step tutorial on using the Authz component can be found on the caGrid wiki here: http://www.cagrid.org/mwiki/index.php?title=CaGrid:How-To:IntegrateCSMAuthorizationPolicy

JAAS Configuration

No changes are required to be made to CSM's JAAS configuration.

ApplicationSecurityConfig.xml

caCORE services that are using CSM will have configured the `gov.nih.nci.security.configFile` system property to point to an `ApplicationSecurityConfig.xml` file. To use the Authz component, specify `gov.nih.nci.cagrid.authorization.CSMGridAuthorizationManager` as the implementation to use for both the authorization manager.

Figure 6-48 contains an example `ApplicationSecurityConfig.xml` file.

```
<security-config>

    <upt-context-name>UPT</upt-context-name>

    <application-list>
```

```
      <application>
         <context-name>SDK</context-name>
         <authentication>
            <lockout-time>100</lockout-time>
            <allowed-login-time>100</allowed-login-time>
            <allowed-attempts>3</allowed-attempts>
            <authentication-provider-class>
            </authentication-provider-class>
         </authentication>
         <authorization>
            <authorization-provider-class>
            gov.nih.nci.cagrid.authorization.impl.CSMGridAuthorizationManager
            </authorization-provider-class>
            <hibernate-config-file>
               /my/app/etc/hibernate.cfg.xml
            </hibernate-config-file>
         </authorization>
      </application>
   </application-list>
</security-config>
```

Figure 6-48 `ApplicationSecurityConfig.xml` file

hibernate.cfg.xml

Hibernate must be configured to use the c3p0 connection pool. Add the following properties to the *session-factory* element in the `hibernate.cfg.xml`file.

```
<property name="hibernate.c3p0.min_size">5</property>
<property name="hibernate.c3p0.max_size">20</property>
<property name="hibernate.c3p0.timeout">300</property>
<property name="hibernate.c3p0.max_statements">50</property>
<property name="hibernate.c3p0.idle_test_period">3000</property>
```

Web Applications Classpath

Table 6-5 contains the jars that must be added to the web applications classpath.

From the Globus 4.0.3 WS Core Distribution	**From other caGrid 1.1 Projects (these end up in cagrid-1-1/ext/lib, when building the Authz project).**	**From the Authz project's lib folder**
addressing-1.0.jar	caGrid-1.1-core.jar	c3p0-0.8.5.2.jar
axis.jar	caGrid-1.1-gridca.jar	clm.jar
cog-axis.jar	caGrid-1.1-gridgrouper-client.jar	csmapi.jar
cog-jglobus.jar	caGrid-1.1-gridgrouper-common.jar	hibernate-3.0.5.jar
cryptix-asn1.jar	caGrid-1.1-gridgrouper-stubs.jar	spring-core.jar
cryptix.jar	caGrid-1.1-metadata-common.jar	spring-beans.jar
cryptix32.jar	caGrid-1.1-metadata-security.jar	
jce-jdk13-125.jar	caGrid-1.1-ServiceSecurityProvider-client.jar	
jgss.jar	caGrid-1.1-ServiceSecurityProvider-common.jar	
puretls.jar	caGrid-1.1-ServiceSecurityProvider-service.jar	
wsrf_common.jar	caGrid-1.1-ServiceSecurityProvider-stubs.jar	
wsrf_core_stubs.jar	cglib-nodep-2.1_3.jar	
wsrf_core.jar	grouper.jar	
wss4j.jar	mobius_common_client.jar	
	mobius_factories.jar	
	mobius_gme_client.jar	
	mobius_mako_client.jar	
	mobius_tools.jar	
	subject-0.2.1.jar	

Table 6-5 jars to add to the web applications classpath

Finally, a file named `ObjectStateLoggerConfig.xml` must be added to the classpath. That file should look like Figure 6-49.

```
<?xml version="1.0" encoding="UTF-8"?>

<logging-config>

   <logger-name>CSM.Audit.Logging.ObjectState.Authorization</logger-name>

   <logger-config-file>log4jConfig.xml</logger-config-file>

   <log-level>info</log-level>

   <messageType>string</messageType>

   <domainObjectList>

      <object-
name>gov.nih.nci.security.authorization.domainobjects.Application</object-name>

      <object-
name>gov.nih.nci.security.authorization.domainobjects.ApplicationContext</object-name>

      <object-name>gov.nih.nci.security.authorization.domainobjects.Group</object-
name>

      <object-
name>gov.nih.nci.security.authorization.domainobjects.GroupRoleContext</object-name>

      <object-
name>gov.nih.nci.security.authorization.domainobjects.Privilege</object-name>

      <object-
name>gov.nih.nci.security.authorization.domainobjects.ProtectionElement</object-name>

      <object-
name>gov.nih.nci.security.authorization.domainobjects.ProtectionElementPrivilegeContex
t</object-name>

      <object-
name>gov.nih.nci.security.authorization.domainobjects.ProtectionGroup</object-name>

      <object-
name>gov.nih.nci.security.authorization.domainobjects.ProtectionGroupRoleContext</obje
ct-name>

      <object-name>gov.nih.nci.security.authorization.domainobjects.Role</object-
name>

      <object-name>gov.nih.nci.security.authorization.domainobjects.User</object-
name>

      <object-
name>gov.nih.nci.security.authorization.domainobjects.UserGroupRoleProtectonGroup</obj
ect-name>

      <object-
name>gov.nih.nci.security.authorization.domainobjects.UserProtectionElement</object-
name>
       <object-
name>gov.nih.nci.security.authorization.domainobjects.UserRoleContext</object-name>
       <object-
name>gov.nih.nci.security.authorization.dao.hibernate.ProtectionGroupProtectionElement
</object-name>
       <object-
name>gov.nih.nci.security.authorization.dao.hibernate.RolePrivilege</object-name>
       <object-
name>gov.nih.nci.security.authorization.dao.hibernate.UserGroup</object-name>
   </domainObjectList>
   <loggingEnabled>true</loggingEnabled>
</logging-config>
```

Figure 6-49 `ObjectStateLoggerConfig.xml`

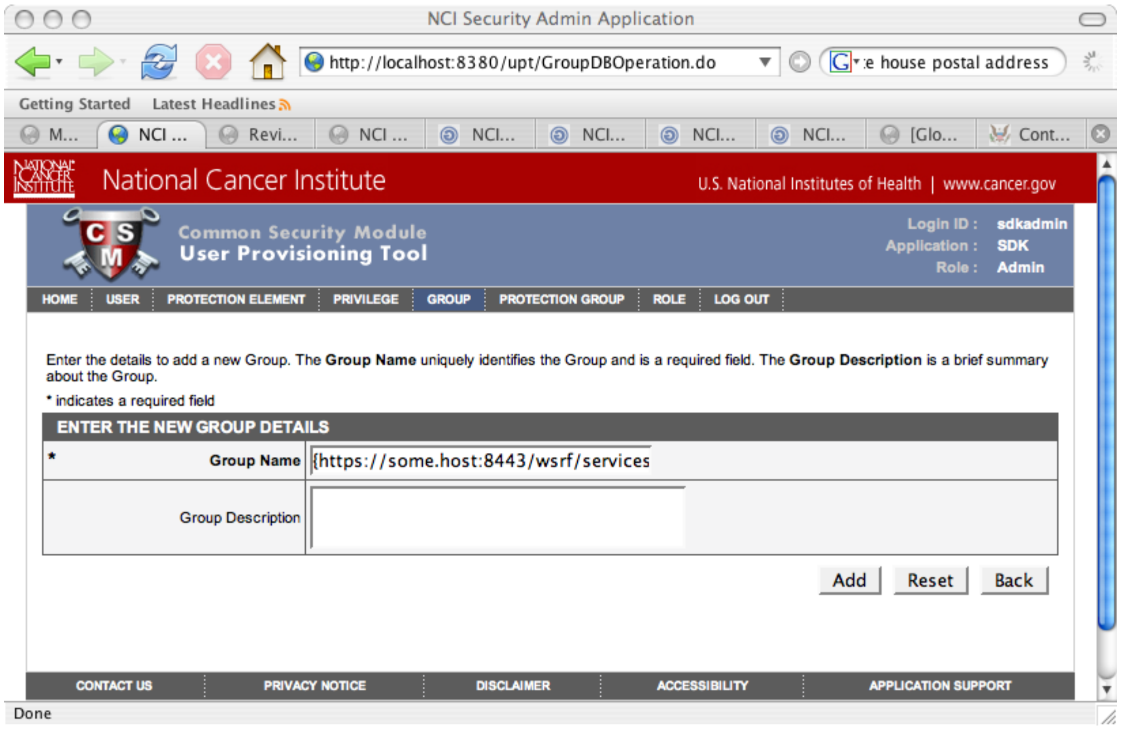
CSM Administration

This section provides an example scenario to illustrate the steps that a local CSM administrator would take to extend access privileges to caGrid users.

In this scenario, the administrator would like to permit members of the "cabig:researchers" group to query his caCORE system for gov.nih.nci.cabio.domain.Gene objects. The Grid Grouper instance in which this group is defined is running at https://some.host:8443/wsrf/services/cagrid/GridGrouper.

Perform the following steps with the UPT.

1. Create a group named {https://some.host:8443/wsrf/services/cagrid/GridGrouper}cabig:researchers (Figure 6-50).



Enter the details to add a new Group. The **Group Name** uniquely identifies the Group and is a required field. The **Group Description** is a brief summary about the Group.

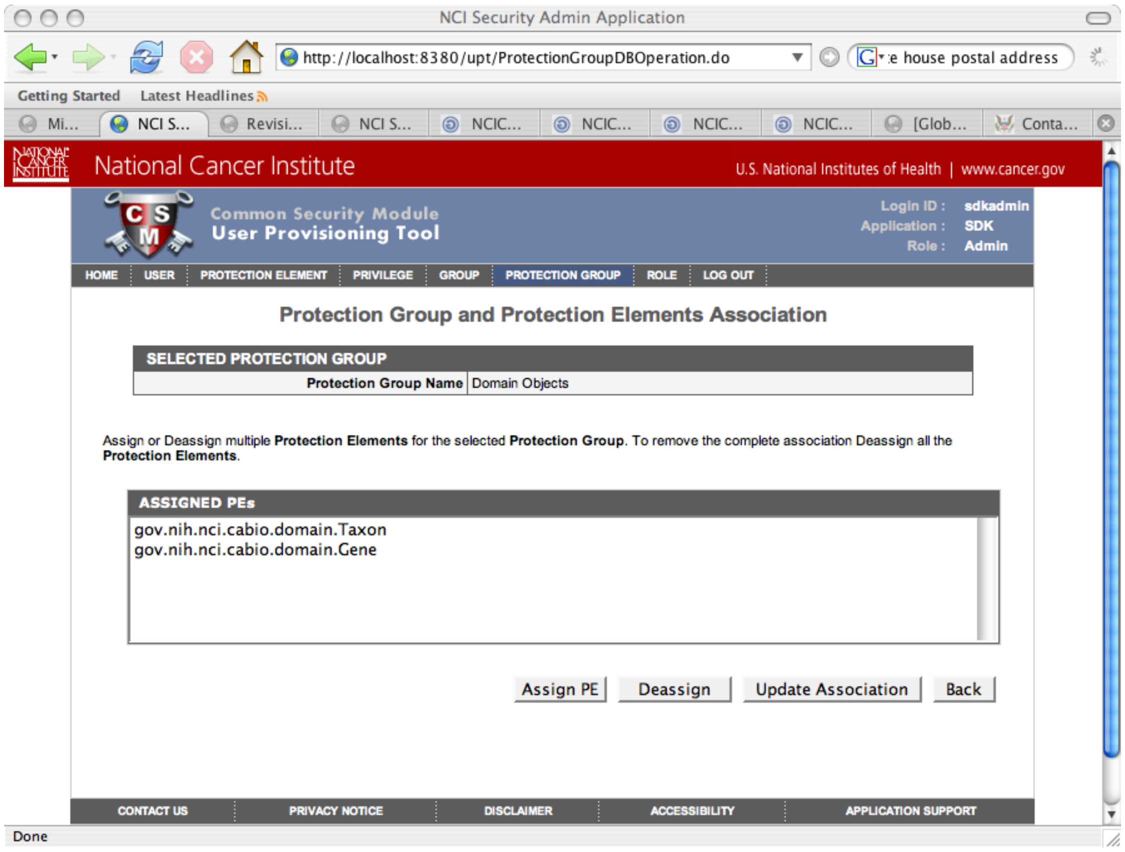
* indicates a required field

ENTER THE NEW GROUP DETAILS

* **Group Name**	{https://some.host:8443/wsrf/services
Group Description	

Figure 6-50 Create a group in the UPT

In this scenario, it is assumed that the “gov.nih.nci.cabio.domain.Gene” protection element already exists, and is a member of the “Domain Objects” protection group (Figure 6-51).



Login ID : sdkadmin
Application : SDK
Role : Admin

Protection Group and Protection Elements Association

SELECTED PROTECTION GROUP

Protection Group Name	Domain Objects

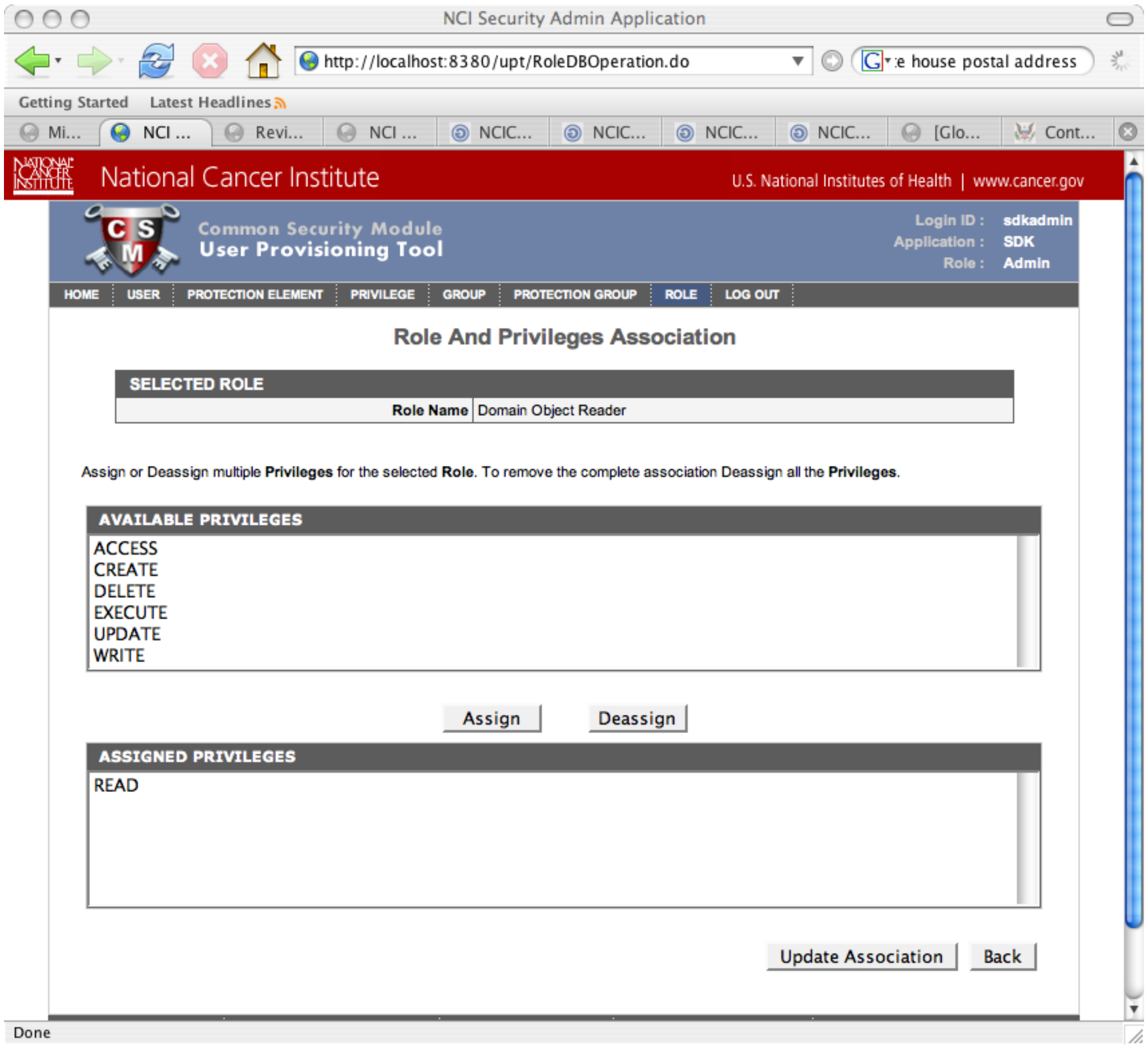
Assign or Deassign multiple **Protection Elements** for the selected **Protection Group**. To remove the complete association Deassign all the **Protection Elements**.

ASSIGNED PEs

gov.nih.nci.cabio.domain.Taxon
gov.nih.nci.cabio.domain.Gene

Figure 6-51 Protection groups and protection group elements in the UPT

2. It is also assumed that a role named "Domain Object Readers" exists and has a single "READ" privilege (Figure 6-52).



Login ID : sdkadmin
Application : SDK
Role : Admin

Role And Privileges Association

SELECTED ROLE

Role Name: Domain Object Reader

Assign or Deassign multiple **Privileges** for the selected **Role**. To remove the complete association Deassign all the **Privileges**.

AVAILABLE PRIVILEGES

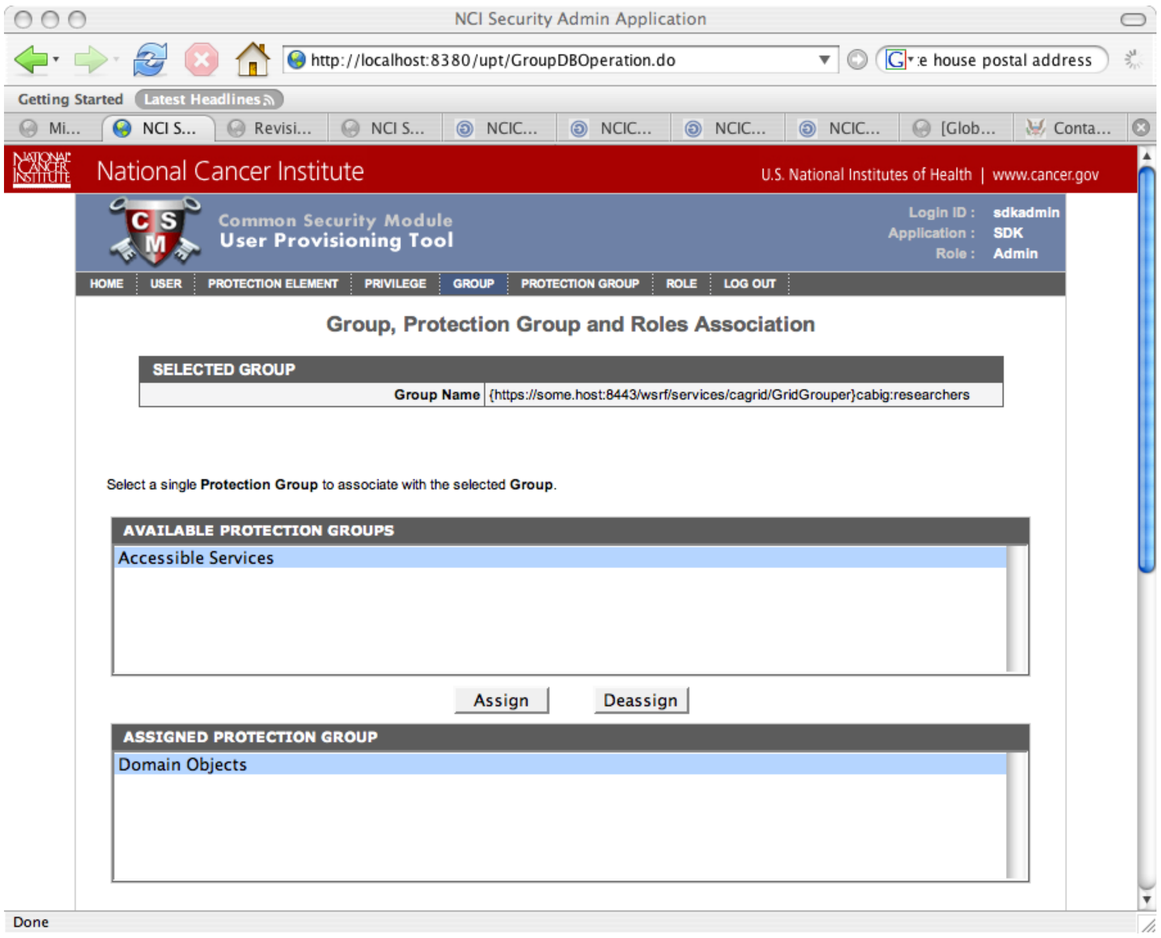
ACCESS
CREATE
DELETE
EXECUTE
UPDATE
WRITE

ASSIGNED PRIVILEGES

READ

Figure 6-52 Role and privileges association in the UPT

3. To grant the “READ” privilege to members of the “{https://some.host:8443/wsrf/services/cagrid/GridGrouper}cabig:researchers” group, assign the “Domain Objects” protection group and “Domain Object Readers” role to this group (Figure 6-53 and Figure 6-54).



Group, Protection Group and Roles Association

SELECTED GROUP

Group Name {https://some.host:8443/wsrf/services/cagrid/GridGrouper}cabig:researchers

Select a single **Protection Group** to associate with the selected **Group**.

AVAILABLE PROTECTION GROUPS

Accessible Services

ASSIGNED PROTECTION GROUP

Domain Objects

Figure 6-53 Group, Protection Group and Roles Association in the UPT

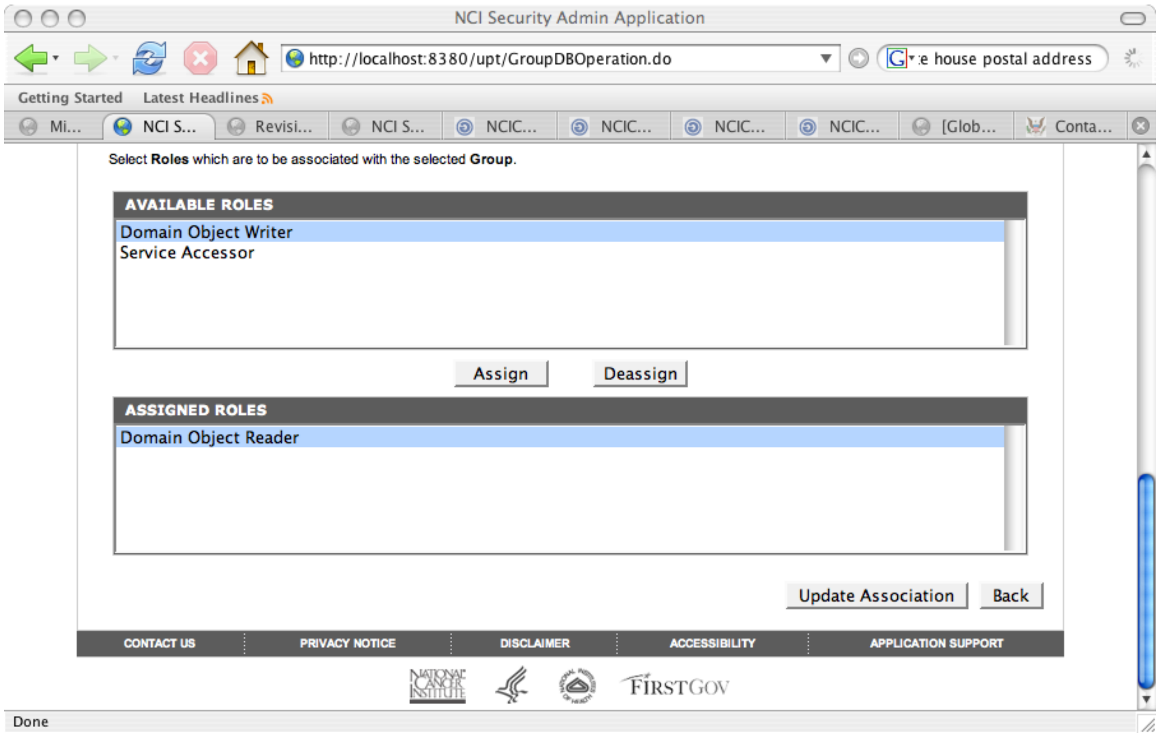


Figure 6-54 Group, Protection Group and Roles Association in the UPT

4. Now, assuming that the following `/O=NIH/OU=NCI/OU=NCICB/OU=DEV/OU=localhost/OU=IdP [1]/CN=george` identity is a member of the "cabig:researchers" group, the following code should print "Authorized: true" (Figure 6-55).

```
String identity =
"/O=NIH/OU=NCI/OU=NCICB/OU=DEV/OU=localhost/OU=IdP [1]/CN=george";
String app = "myapp";
String objectId = "gov.nih.nci.cabio.domain.Gene";
String privilege = "READ";
AuthorizationManager mgr =
SecurityServiceProvider.getAuthorizationManager(app);
boolean authorized = mgr.checkPermission(identity, objectId, privilege);
System.out.println("Authorized: " + authorized);
```

Figure 6-55 Code verification

Chapter 7 Workflow Services

This chapter describes the caGrid implementation of a workflow, which provides a grid service for submitting and running workflows that are composed of other grid services.

Topics in this chapter include:

Introduction

caBIG aims to bring together disparate data and analytic resources into a “World Wide Web of cancer research”. This will be achieved through common standards and software frameworks for the federation of these resources into “grid” services. Many of the tasks in the collection and analysis of cancer-related data on the grid involve the use of workflow. In this context, workflow is defined as the connecting of services to solve a problem that each individual service could not solve. caGrid implements workflow by providing a grid service for submitting and running workflows that are composed of other grid services.

The Business Process Execution Language (BPEL)

The Business Process Execution Language (BPEL) is an XML language for describing business process behavior based on web/grid services. BPEL is layered on top of other Web technologies such as WSDL 1.1, XML Schema 1.0, XPath 1.0, and WS Addressing, which makes it a perfect candidate for use in caGrid. The BPEL notation includes flow control, variables, concurrent execution, input and output, transaction scoping/compensation, and error handling. A BPEL *process* describes a business process, which often invokes Web/Grid services to perform functional tasks. A process can be either *abstract* or *executable*. Abstract processes are similar to library APIs: they describe what the process can do with inputs and outputs, but they do not describe how the work actually gets done. Abstract processes are useful for describing a business process to another party that wants to implement the process. Executable processes do the "heavy lifting"; that is, they contain all of the execution steps that represent a cohesive unit of work. The focus of this document is on executable processes, as they are concrete workflows that can run through the workflow service.

Some vocabulary must be established to understand a BPEL document. While a typical domain user such as an oncologist is not expected to write a BPEL document, it is expected that developers will be able to produce BPEL from higher-level tools. In BPEL, a process consists of *activities* connected by *links*. A process sometimes only contains one activity, but that is usually a container for more activities. The path taken through the activities and their links is determined by many things, including the values of variables and the evaluation of expressions. The starting points are called *start activities*, and their "create instance" attributes are set to "yes". When a start activity is triggered, a new business process instance is created. Each service that is invoked by the workflow is called a *PartnerLink,* and BPEL extends this concept to include the client that is invoking the workflow.

Creating a Sample Workflow Using Test Services

A general process for creating and submitting workflows using caGrid can be defined by the following high level steps:

1. Get the endpoints of the services of which the workflow should consist. These endpoints can be obtained from a query to the Index Service based on the Service Metadata, though that must be done prior to creating the workflow.
2. Define Partner Links for the services that will interact.
3. Create a BPEL document using a GUI if available.
4. Submit the BPEL document to the WorkflowFactoryService using the Workflow GUI client.
5. The command-line client submits the workflow and starts it using the specified input document.

Installing Test Services

To test the workflow service locally, a set of simple test services are provided, and can be installed using the following steps:

1. Download the services from http://gforge.nci.nih.gov/frs/download.php/2223/workflow-services.zip
2. Unzip the services into a directory.
3. Enter `cd workflow-services/TestService1`
4. Make sure that $CATALINA_HOME is set and points to a working Tomcat installed with caGrid 1.1.
5. Run ant deployTomcat. This deploys the Test Service1 in Tomcat. Do not install the second service yet.
6. Restart Tomcat and check if the service is up by testing this link : http://<hostname>:<port>/wsrf/services/cagrid/WorkflowTestService1?wsdl

Configuring and Running a Workflow

The Workflow Submission GUI is for submitting and monitoring BPEL workflows. It also allows discovering and adding services to be used in the workflow and reports the output of the workflow once the workflow has executed.

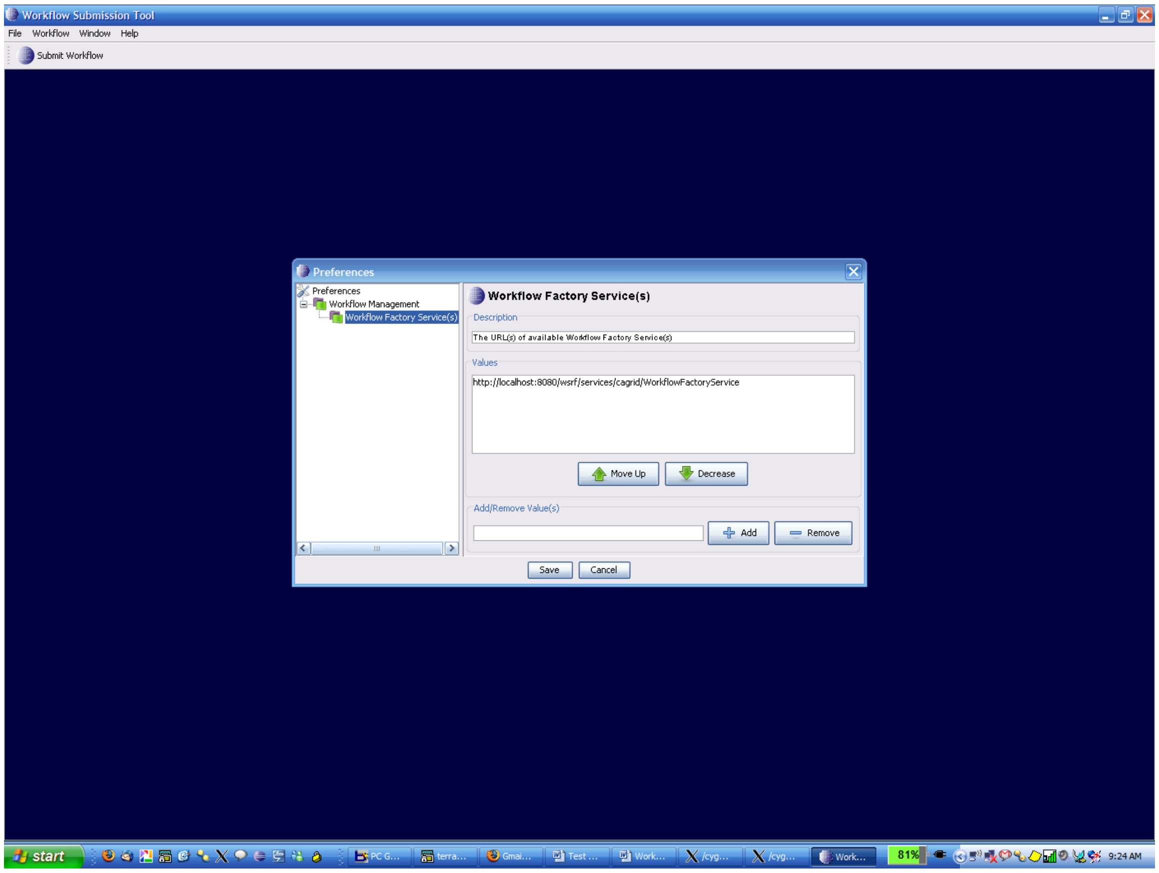
Note: Prerequisites to using the Workflow Submission GUI are a Tomcat container with caGrid installed and a Tomcat container with ActiveBPEL installed. These are already done if the Workflow component was installed using the caGrid installer.

The following sections describe the capabilities of the Workflow Submission GUI, by using the example workflow and services provided by caGrid that were created and installed above.

Launching the Workflow Submission GUI

Use the following steps to launch the Workflow Submission GUI.

2. Browse to the WorkflowFactoryService dir under the caGrid source distribution (caGrid/projects/workflow/WorkflowFactoryService/). From this directory, run `ant ui` to launch the Workflow GUI.
3. In the GU, select **Window -> Preferences**. Browse to the WorkflowFactoryService(s) endpoint option and make sure it points to the Workflow service that needs to be validated (Figure 7-1).
4. Use the **Add** and **Remove** buttons to add new Workflow services endpoints. Use the **Move up** and **Decrease** keys to move the endpoint up and down.



Description

The URL(s) of available Workflow Factory Service(s)

Values

http://localhost:8080/wsrf/services/cagrid/WorkflowFactoryService

Add/Remove Value(s)

Figure 7-1 Adding service endpoints in the Workflow Submission GUI

Submitting a Workflow

To submit a workflow, use the following steps.

1. Click **Submit Workflow** to open the Submit Workflow window. Enter the path to the BPEL document in the BPEL File text field or browse to the location of the BPEL document (Figure 7-2). For the validation test, select the workflow document **Test1.bpel** in the WorkflowFactoryService folder.

2. Name the workflow name the same as the name of the BPEL document (Test1 in the validation example) and click **Add Partner Links**.

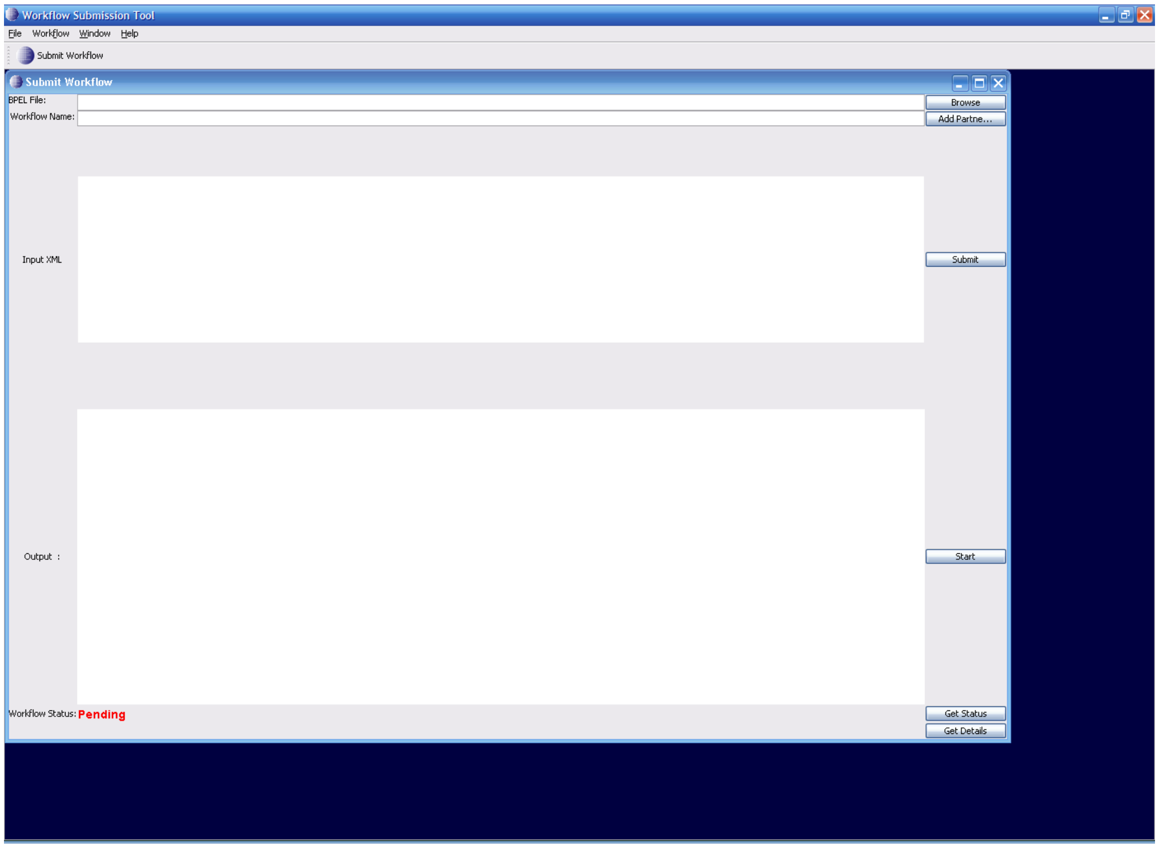


Figure 7-2 Submit Workflow window

3. In the Partner Link Frame dialog that opens, enter the endpoints of the services included in the workflow (Figure 7-3). In the validation example, there is one service invoked twice in the workflow. Enter appropriate values in the fields. For the validation test, the values are:

   Select Type: Static

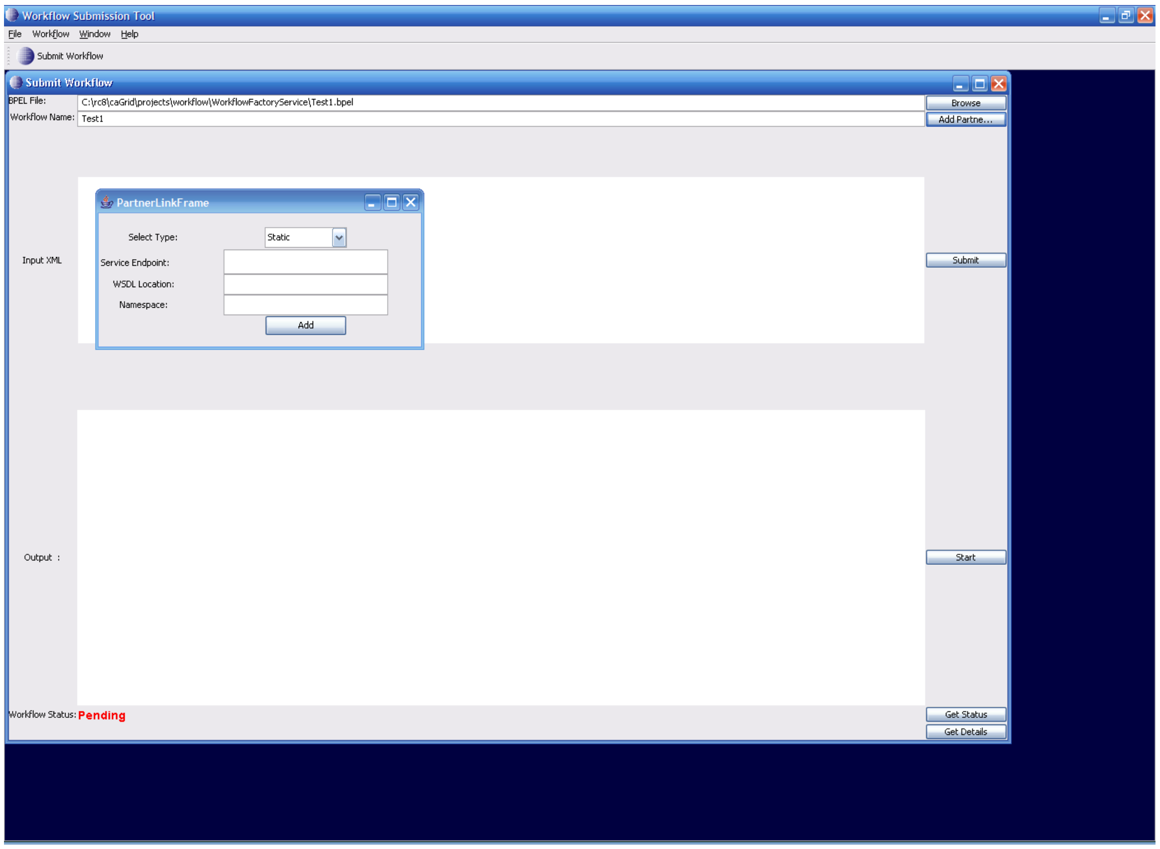
   Service Endpoint: http://localhost:8080/wsrf/services/cagrid/WorkflowTestService1

   WSDL Location: http://localhost:8080/wsrf/share/schema/WorkflowTestService1/WorkflowTestService1.wsdl

   Namespace: http://sample1.tests.workflow.cagrid.nci.nih.gov/WorkflowTestService1

   The localhost:8080 should be replaced with the appropriate host:port on which the test service is deployed (From the first part of the document).

4. Click **Add**.

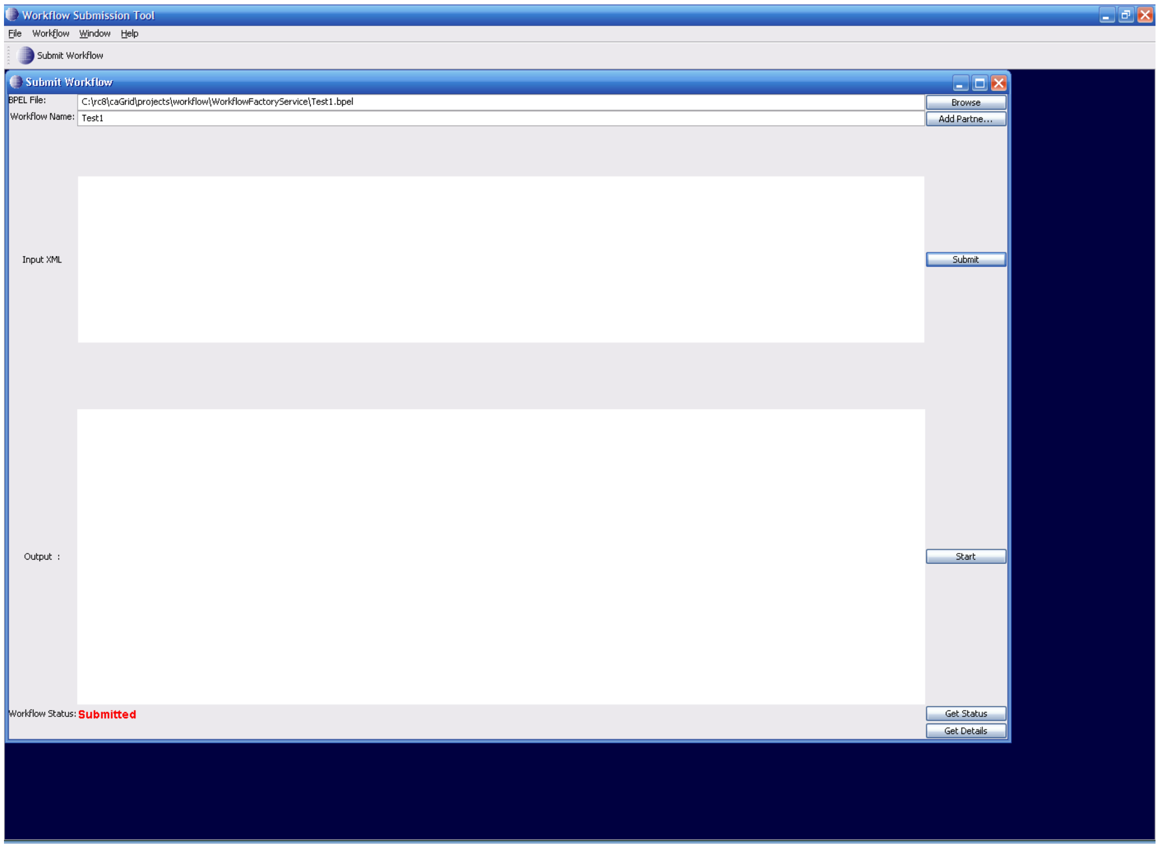


BPEL File: C:\rc8\caGrid\projects\workflow\WorkflowFactoryService\Test1.bpel
Workflow Name: Test1
Select Type: Static
Service Endpoint:
WSDL Location:
Namespace:
Workflow Status: Pending

Figure 7-3 Partner Link Frame dialog

Submitting a Workflow

Click **Submit** to submit the workflow to a pre-configured Workflow Factory Service. The BPEL document is then validated and submitted to the Workflow Engine. If there are no errors the Status is changed from Pending to Submitted (Figure 7-4).



BPEL File: C:\rc8\caGrid\projects\workflow\WorkflowFactoryService\Test1.bpel
Workflow Name: Test1
Workflow Status: Submitted

Figure 7-4 Submitting a workflow

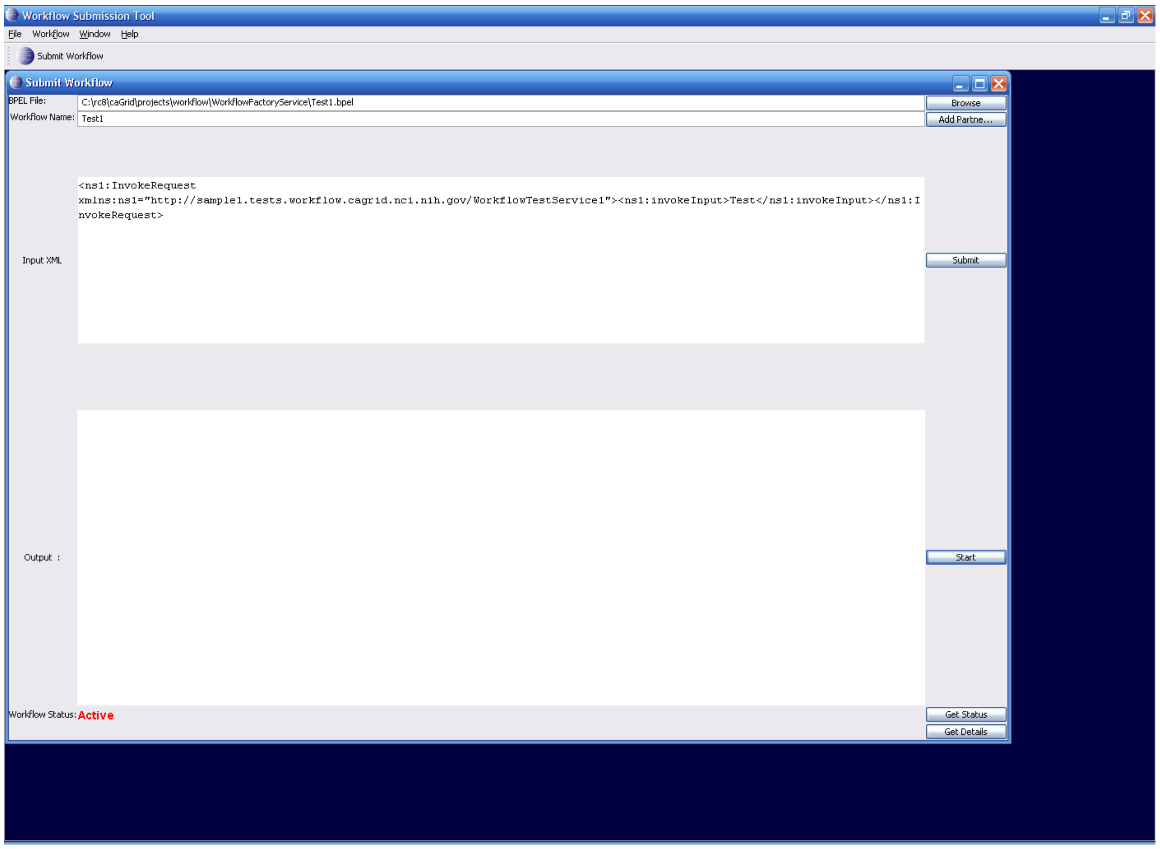
Executing a Workflow

The next step is to execute the submitted workflow with some input.

1. In the current implementation, the input XML must be pasted into the text area. For the validation test, copy the following XML blob into the input XML Text area:

   <ns1:InvokeRequest xmlns:ns1="http://sample1.tests.workflow.cagrid.nci.nih.gov/WorkflowTestService1"><ns1:invokeInput>Test</ns1:invokeInput></ns1:InvokeRequest>

2. Click **Start**. The workflow starts and the status changes from Submitted to Active (Figure 7-5).



BPEL File: C:\rc8\caGrid\projects\workflow\WorkflowFactoryService\Test1.bpel

Workflow Name: Test1

```
<ns1:InvokeRequest
xmlns:ns1="http://sample1.tests.workflow.cagrid.nci.nih.gov/WorkflowTestService1"><ns1:invokeInput>Test</ns1:invokeInput></ns1:I
nvokeRequest>
```

Workflow Status: Active

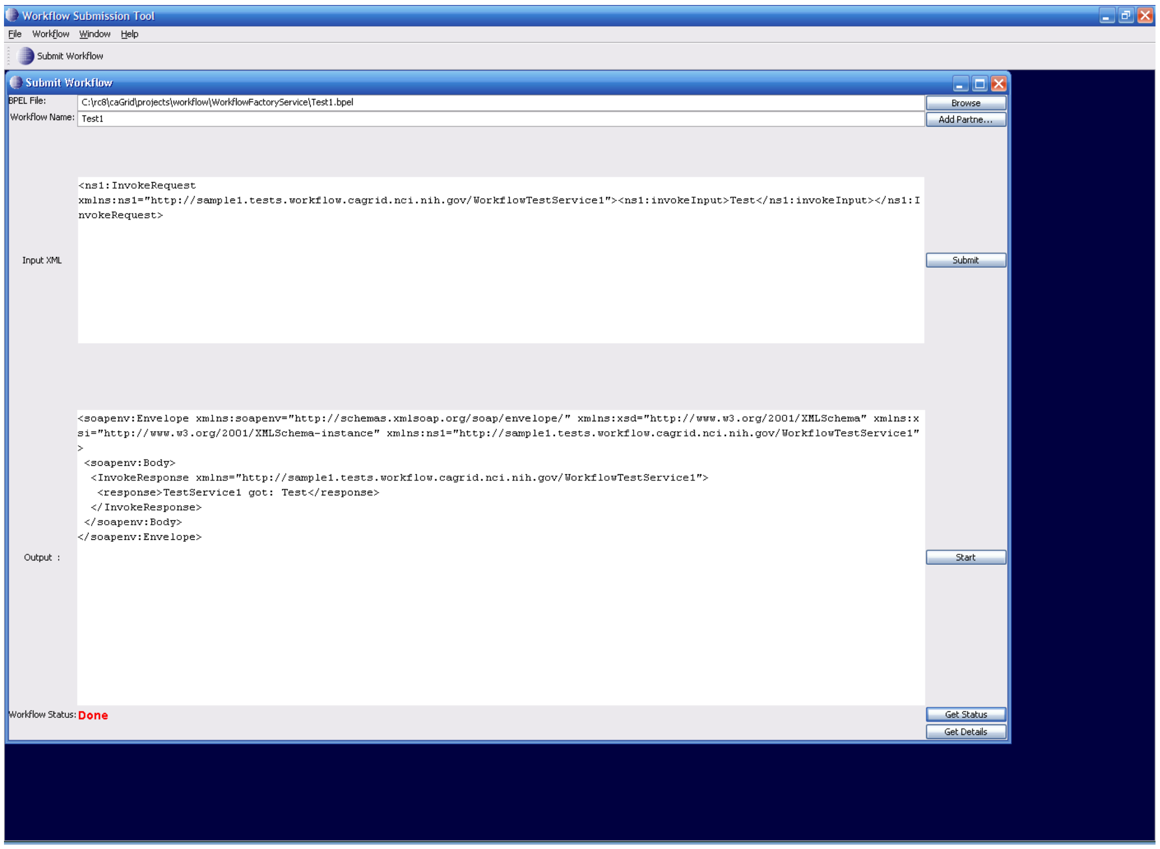
Figure 7-5 Executing a workflow

Querying for Status

For workflow status, click **Get Status**. If the status is different from the existing status (on the left hand corner of the GUI), it updates to the latest status.

Getting Workflow Output

For workflow output, click **Get Status**. If the workflow is **Done**, the workflow output is displayed in the *output* text area (Figure 7-6).



BPEL File: C:\rc8\caGrid\projects\workflow\WorkflowFactoryService\Test1.bpel

Workflow Name: Test1

Input XML

```
<ns1:InvokeRequest
xmlns:ns1="http://sample1.tests.workflow.cagrid.nci.nih.gov/WorkflowTestService1"><ns1:invokeInput>Test</ns1:invokeInput></ns1:I
nvokeRequest>
```

Output :

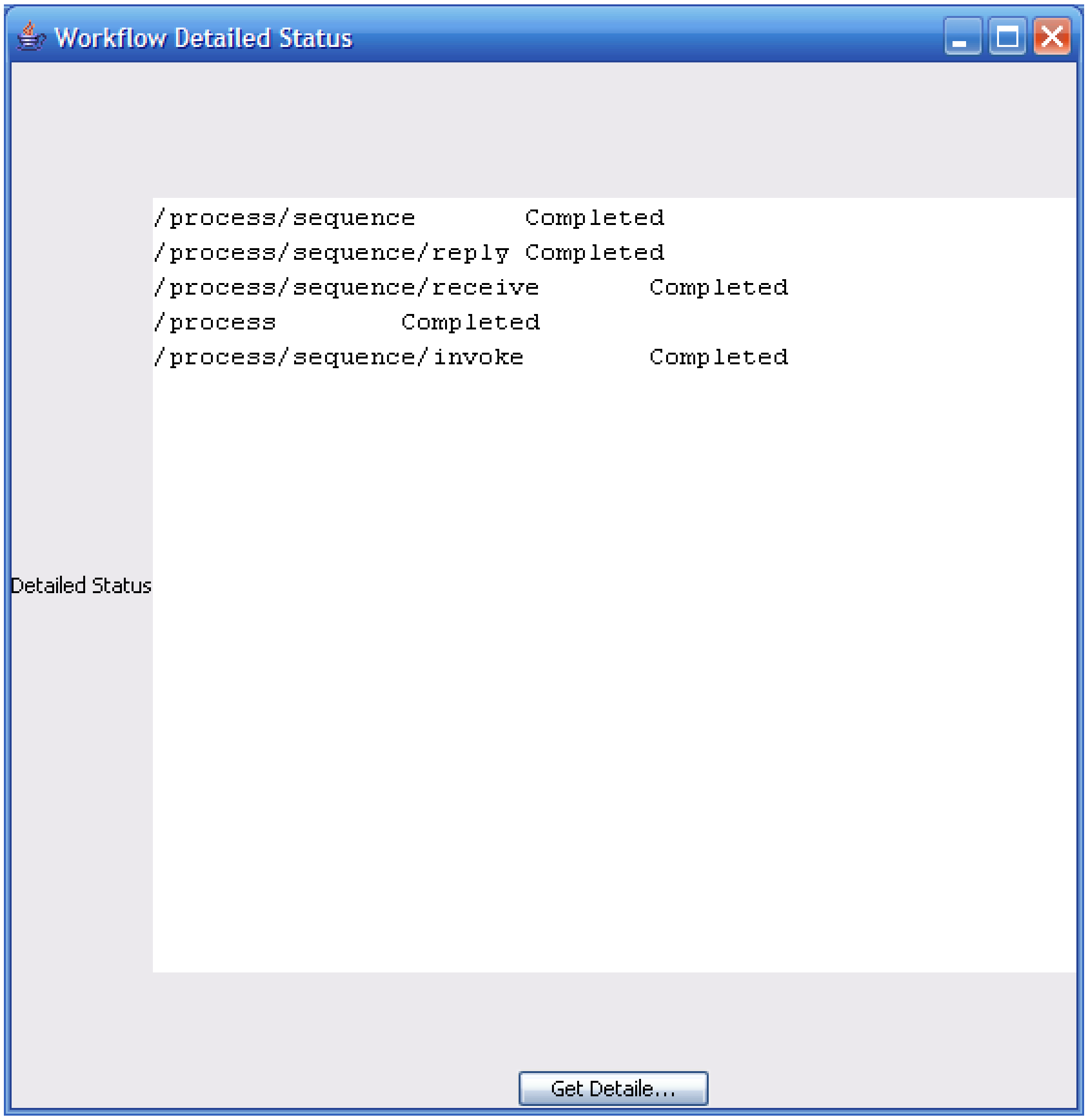
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:x
si="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns1="http://sample1.tests.workflow.cagrid.nci.nih.gov/WorkflowTestService1"
>
 <soapenv:Body>
  <InvokeResponse xmlns="http://sample1.tests.workflow.cagrid.nci.nih.gov/WorkflowTestService1">
   <response>TestService1 got: Test</response>
  </InvokeResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Workflow Status: Done

Figure 7-6 Workflow output

Getting Detailed Status

When a long workflow is submitted and you would like to see which portion of the workflow is currently being executed, click **Get Details** to see where the current workflow execution is taking place. The GUI displays an XPath expression of the node in Workflow Document and its status (Figure 7-7)



```
/process/sequence        Completed
/process/sequence/reply Completed
/process/sequence/receive        Completed
/process        Completed
/process/sequence/invoke        Completed
```

Figure 7-7 Details status

Terminating a Workflow

Since workflows are modeled as WSRF resources, they have a lifetime associated with them. The Workflow service provides a standard "destroy" operation to stop the workflow and free up all the resources that are used by the workflow

Pausing and Resuming a Workflow

The command-line client provides an operation by which an active workflow can be paused. Running this command results in the service invoking the "pause" operation of the workflow management service using the workflow id.

The command-line client also provides an operation by which a paused workflow can be resumed. Running this command results in the service invoking the “resume” operation of the workflow management service using the workflow id.

Appendix A References

Scientific Publications

[1] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and T. S., "Data Management and Transfer in High Performance Computational Grid Environments," *Parallel Computing Journal*, vol. 28, pp. 749-771, 2002.

[2] W. E. Allcock, I. Foster, and R. Madduri, "Reliable Data Transport: A Critical Service for the Grid.," in *Proceedings of Building Service Based Grids Workshop, Global Grid Forum 11*. Honolulu, Hawaii, USA, 2004.

[3] G. Allen, T. Dramlitsch, I. Foster, T. Goodale, N. Karonis, M. Ripeanu, E. Seidel, and B. Toonen, "Cactus-G Toolkit: Supporting Efficient Execution in Heterogeneous Distributed Computing Environments," in *Proceedings of the 4th Globus Retreat*. Pittsburg, PA, 2000.

[4] H. Andrade, T. Kurc, A. Sussman, and J. Saltz, "Active Proxy-G: Optimizing the Query Execution Process in the Grid," in *Proceedings of the ACM/IEEE Supercomputing Conference (SC2002)*. Baltimore, MD: ACM Press/IEEE Computer Society Press, 2002.

[5] J. Annis, Y. Zhao, J. Voeckler, M. Wilde, S. Kent, and I. Foster, "Applying Chimera Virtual Data Concepts to Cluster Finding in the Sloan Sky Survey," in *Proceedings of the ACM/IEEE Supercomputing Conference (SC2002)*. Baltimore, MD: ACM Press/IEEE Computer Society Press, 2002.

[6] M. P. Atkinson and et.al., "Grid Database Access and Integration: Requirements and Functionalities," Technical Document, Global Grid Forum. http://www.cs.man.ac.uk/grid-db/documents.html, 2002.

[7] F. Berman, H. Casanova, J. Dongarra, I. Foster, C. Kesselman, J. Saltz, and R. Wolski, "Retooling Middleware for Grid Computing," *NPACI & SDSC enVision*, vol. 18, 2002.

[8] M. Beynon, T. Kurc, A. Sussman, and J. Saltz, "Design of a Framework for Data-Intensive Wide-Area Applications," in *Proceedings of the 2000 Heterogeneous Computing Workshop (HCW2000)*. Cancun, Mexico, 2000.

[9] H. Casanova, O. Graziano, F. Berman, and R. Wolski, "The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid," in *Proceedings of the ACM/IEEE Supercomputing Conference (SC2000)*: ACM Press/IEEE Computer Society Press, 2000.

[10] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B. Schwartzkopf, H. Stockinger, and B. Tierney, "Giggle: A Framework for Constructing Scalable Replica Location Services," in *Proceedings of the*

ACM/IEEE Supercomputing Conference (SC2002): ACM Press/IEEE Computer Computer Society Press, 2002, pp. 1-17.

[11] A. Chervenak, E. Deelman, C. Kesselman, B. Allcock, I. Foster, V. Nefedova, J. Lee, A. Sim, A. Shoshahi, B. Drach, D. Williams, and D. Middleton, "High-performance remote access to climate simulation data: a challenge problem for data grid technologies," *Parallel Computing*, vol. 29, pp. 1335-1356, 2003.

[12] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," *Journal of Network and Computer Applications*, vol. 23, pp. 187-200, 2000.

[13] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda, "Mapping Abstract Complex Workflows onto Grid Environments," *Journal of Grid Computing*, vol. 1, pp. 25-39, 2003.

[14] E. Deelman, G. Singh, M. P. Atkinson, A. Chervenak, N. P. Chue Hong, C. Kesselman, S. Patil, L. Pearlman, and M. Su, "Grid-Based Metadata Services," in *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM '04)*, 2004.

[15] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit.," *International Journal of High Performance Computing Applications*, vol. 11, pp. 115--128, 1997.

[16] I. Foster, J. Voeckler, M. Wilde, and Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation," in *Proceedings of the 14th Conference on Scientific and Statistical Database Management (SSDBM '02)*, 2002.

[17] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: A Computational Management Agent for Multi-institutional Grids," in *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*: IEEE Press, 2001.

[18] N. Furmento, W. Lee, A. Mayer, S. Newhouse, and J. Darlington, "ICENI: An Open Grid Service Architecture Implemented with JINI," in *Proceedings of the ACM/IEEE Supercomputing Conference (SC2002)*. Baltimore, MD: ACM Press/IEEE Computer Society Press, 2002.

[19] A. S. Grimshaw and W. Wulf, "The Legion: Vision of a Worldwide Virtual Computer," *Communications of the ACM*, vol. 40, pp. 39--45, 1997.

[20] S. Hastings, S. Langella, S. Oster, and J. Saltz, "Distributed Data Management and Integration: The Mobius Project," *Proceedings of the Global Grid Forum 11 (GGF11) Semantic Grid Applications Workshop, Honolulu, Hawaii, USA.*, pp. 20-38, 2004.

[21] S. Langella, S. Oster, S. Hastings, F. Siebenlist, T. Kurc, and J. Saltz, "Dorian: Grid Service Infrastructure for Identity Management and Federation," presented at The 19th IEEE Symposium on Computer-Based Medical Systems, Special Track: Grids for Biomedical Informatics, Salt Lake City, Utah., 2006.

[22] R. Oldfield and D. Kotz, "Armada: A Parallel File System for Computational Grid," in

Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid2001). Brisbane, Australia: IEEE Computer Society Press, 2001.

[23] M. Sato, H. Nakada, S. Sekiguchi, S. Matsuoka, U. Nagashima, and H. Takagi, "Ninf: A Network based Information Library for a Global World-Wide Computing Infrastructure," in *Proceedings of the Conference on High Performance Computing and Networking (HPCN '97) (LNCS-1225)*, 1997, pp. 491-502.

[24] G. Singh, S. Bharathi, A. Chervenak, E. Deelman, C. Kesselman, M. Mahohar, S. Pail, and L. Pearlman, "A Metadata Catalog Service for Data Intensive Applications," in *Proceedings of the ACM/IEEE Supercomputing Conference (SC2003)*, 2003.

[25] G. Singh, E. Deelman, G. Mehta, K. Vahi, M. Su, B. Berriman, J. Good, J. Jacob, D. Katz, A. Lazzarini, K. Blackburn, and S. Koranda, "The Pegasus Portal: Web Based Grid Computing," in *Proceedings of the 20th Annual ACM Symposium on Applied Computing*. Santa Fe, New Mexico, 2005.

[26] J. Smith, A. Gounaris, P. Watson, N. W. Paton, A. A. Fernandes, and R. Sakellariou, "Distributed Query Processing on the Grid.," presented at Proceedings of the Third Workshop on Grid Computing (GRID2002), Baltimore, MD, 2003.

[27] D. Thain, J. Basney, S. Son, and M. Livny, "Kangaroo Approach to Data Movement on the Grid," in *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC-10)*, 2001.

[28] L. Weng, G. Agrawal, U. Catalyurek, T. Kurc, S. Narayanan, and J. Saltz, "An Approach for Automatic Data Virtualization," in *Proceedings of the 13th IEEE International Symposium on High-Performance Distributed Computing (HPDC-13)*. Honolulu, Hawaii, 2004, pp. 24-33.

[29] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Open Grid Service Infrastructure Working Group Technical Report, Global Grid Forum. http://www.globus.org/alliance/publications/papers/ogsa.pdf 2002.

[30] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations.," *International Journal of Supercomputer Applications*, vol. 15, pp. 200-222, 2001.

[31] E. Cerami, *Web Services Essentials*: O'Reilly & Associates Inc., 2002.

[32] S. Graham, S. Simeonov, T. Boubez, D. Davis, G. Daniels, Y. Nakamura, and R. Neyama, *Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI*: SAMS Publishing, 2002.

[33] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, "The WS-Resource Framework version 1.0," vol. 2004, 2004.

[34] J. Saltz, S. Oster, S. Hastings, T. Kurc, W. Sanchez, M. Kher, A. Manisundaram, K. Shanbhag, and P. Covitz, "caGrid: Design and Implementation of the Core Architecture

of the Cancer Biomedical Informatics Grid," *Bioinformatics. (in press).* 2006.

[35] S. Langella, S. Hastings, S. Oster, T. Kurc, U. Catalyurek, and J. Saltz, "A Distributed Data Management Middleware for Data-Driven Application Systems," in *Proceedings of the 2004 IEEE International Conference on Cluster Computing (Cluster 2004)*, 2004.

[36] K. Bhatia, S. Chandra, and K. Mueller, "GAMA: Grid Account Management Architecture," San Diego Supercomputer Center (SDSC), UCSD Technical Report. #TR-2005-3, 2005.

[37] I. Foster, C. Kesselman, S. Tuecke, V. Volmer, V. Welch, R. Butler, and D. Engert, "A National Scale Authentication Infrastructure," *IEEE Computer*, vol. 33, pp. 60-66, 2000.

[38] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke, "Security for Grid Services," presented at 12th International Symposium on High Performance Distributed Computing (HPDC-12), 2003.

[39] H. Morohoshi and R. Huang, "A User-friendly Platform for Developing Grid Services over Globus Toolkit 3," presented at The 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05), 2005.

[40] S. Mizuta and R. Huang, "Automation of Grid Service Code Generation with AndroMDA for GT3," presented at The 19th International Conference on Advanced Information Networking and Applications (AINA'05), 2005.

[41] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," *Concurrency and Computation: Practice and Experience*, vol. 13, pp. 643-662, 2001.

[42] G. von Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke, "CoG Kits: A Bridge Between Commodity Distributed Computing and High Performance Grids," presented at ACM Java Grande 2000 Conference, 2000.

[43] R. Buyya and S. Venugopal, "The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report," presented at the First IEEE International Workshop on Grid Economics and Business Models (GECON 2004), New Jersey, USA, 2004.

[44] M. Humphrey and G. Wasson, "Architectural Foundations of WSRF.NET," *International Journal of Web Services Research*, vol. 2, pp. 83-97, 2005.

[45] M. Smith, T. Friese, and B. Freisleben, "Model Driven Development of Service Oriented Grid Applications," presented at Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW '06), 2006.

Technical Manuals/Articles

National Cancer Institute. "caCORE SDK 3.2.1 Programmer's Guide", ftp://ftp1.nci.nih.gov/pub/cacore/SDK/v3.2.1/caCORE_SDK_3.2.1_Programmers_Guide.pdf

National Cancer Institute. "caCORE 3.2 Technical Guide", ftp://ftp1.nci.nih.gov/pub/cacore/caCORE3.2_Tech_Guide.pdf

Java Bean Specification: http://java.sun.com/products/javabeans/docs/spec.html

Foundations of Object-Relational Mapping: http://www.chimu.com/publications/objectRelational/

Object-Relational Mapping articles and products:
http://www.service-architecture.com/object-relational-mapping/
Hibernate Reference Documentation: http://www.hibernate.org/hib_docs/reference/en/html/
Basic O/R Mapping: http://www.hibernate.org/hib_docs/reference/en/html/mapping.html
Java Programming: http://java.sun.com/learning/new2java/index.html
Javadoc tool: http://java.sun.com/j2se/javadoc/
JUnit: http://junit.sourceforge.net/
Extensible Markup Language: http://www.w3.org/TR/REC-xml/
XML Metadata Interchange: http://www.omg.org/technology/documents/formal/xmi.htm
Global Grid Forum: http://www.gridforum.org
Globus: http://www.globus.org
Mobius: http://www.projectmobius.org
W3C: http://www.w3c.org
OGSA-DAI: http://www.ogsadai.org
Apache: http://www.apache.org
Globus Toolkit 3 Programmer's Tutorial:
http://gdp.globus.org/gt3-tutorial/singlehtml/progtutorial_0.4.3.html
XPath tutorial: http://www.w3schools.com/xpath/xpath_syntax.asp
Globus Security Overview:
http://www.ogsadai.org.uk/docs/OtherDocs/SECURITY-FOR-DUMMIES.pdf
High level Overview of Grid:
http://gridcafe.web.cern.ch/gridcafe/index.html
Overview of Globus Toolkit 3 and the OGSI architecture :
http://www-128.ibm.com/developerworks/grid/library/gr-gt3/

caBIG Material

caBIG: http://cabig.nci.nih.gov/
caBIG Compatibility Guidelines: http://cabig.nci.nih.gov/guidelines_documentation

caCORE Material

caCORE: http://ncicb.nci.nih.gov/NCICB/infrastructure
caBIO: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/caBIO
caDSR: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/cadsr

EVS: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/vocabulary

CSM: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/csm

Glossary

Term	*Definition*
API	Application Programming Interface
Authz	caGrid Authorization component
BPEL	Business Process Execution Language
CA	Certificate Authority
caArray	cancer Array Informatics
caBIG	cancer Biomedical Informatics Grid
caBIO	Cancer Bioinformatics Infrastructure Objects
caCORE	cancer Common Ontologic Representation Environment
caDSR	Cancer Data Standards Repository
caGrid	Current test bed architecture of caBIG
CRL	Certificate Revocation List
CSM	Common Security Module
CVS	Concurrent Versions System
DAO	Data Access Objects
DN	Distinguished Name
IdP	Identity Provider
EPR	End Point Reference
EVS	Enterprise Vocabulary Services
GAARDS	Grid Authentication and Authorization with Reliably Distributed Services
GDE	Introduce Graphical Development Environment
GForge	Primary site for collaborative project development for the NCI Center for Bioinformatics (NCICB) and for the NCI's Cancer Biomedical Informatics Grid™ (caBIG)
GGF	Global Grid Forum
GME	Mobius Global Model Exchange - DNS-like service for the universal creation, versioning, and sharing of data descriptions
Grid Service	Basically a Web Services with improved characteristics and standard services like stateful and potentially transient services, Service Data, Notifications, Service Groups, portType extension, and Lifecycle management.

Term	**Definition**
GSH	Grid Service Handle
GSI	Grid Security Infrastructure - represents the latest evolution of the Grid Security Infrastructure. GSI in GT3 builds off of the functionality present in early GT2 toolkit releases - X.509 certificates, TLS/SSL for authentication and message protection, X.509 Proxy Certificates for delegation and single sign-on.
GTS	Grid Trust Service - maintains a federated trust fabric of all the trusted digital signers in the grid
HTTP	Hypertext Transfer Protocol
ISO	International Organization for Standardization
JAAS	Java Authentication and Authorization Service
JAR	Java Archive
Javadoc	Tool for generating API documentation in HTML format from doc comments in source code (http://java.sun.com/j2se/javadoc/)
JDBC	Java Database Connectivity
JUnit	A simple framework to write repeatable tests (http://junit.sourceforge.net/)
LDAP	Lightweight Directory Access Protocol
MAGE	MicroArray and Gene Expression
MAGE-OM	MicroArray Gene Expression - Object Model
Metadata	Definitional data that provides information about or documentation of other data.
MGED	Microarray Gene Expression Data
Mobius	An array of tools and middleware components to coherently share and manage data and metadata in a Grid and/or distributed computing environment.
NCI	National Cancer Institute
NCICB	National Cancer Institute Center for Bioinformatics
OGSA	Open Grid Services Architecture - developed by the Global Grid Forum, aims to define a common, standard, and open architecture for grid-based applications.
OGSI	Open Grid Services Infrastructure -gives a formal and technical specification of what a Grid Service is. In other words, for a high-level architectural view of what Grid Services are, and how they fit into the next generation of grid applications
PKI	Public Key Cryptography
RDBMS	Relational Database Management System

Term	***Definition***
SAML	Secure Access Markup Language
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TRA	Trusted Registration Authority
UI	User Interface
UID	User Identification
UML	Unified Modeling Language
UPT	User Provisioning Tool
URL	Uniform Resource Locators
Virtualization	Make a computational or data resource available to caBIG community - some people call "Gridification"
VO	Virtual Organization
WAR	Web Application Archive
Web Service	Application to application communication using web based service interfaces as describe by the Web Services 1.0 or 2.0 specification.
WSDD	Web Service Deployment Descriptor
WSDL	Web Services Description Language
WSRF	Web Services Resource Framework
X.509 Certificate	With its corresponding private key forms a unique credential or so-called “grid credential” within the grid
XMI	XML Metadata Interchange (http://www.omg.org/technology/documents/formal/xmi.htm) - The main purpose of XMI is to enable easy interchange of metadata between modeling tools (based on the OMG-UML) and metadata repositories (OMG-MOF) in distributed heterogeneous environments
XML	Extensible Markup Language (http://www.w3.org/TR/REC-xml/) - XML is a subset of Standard Generalized Markup Language (SGML). Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML
XPath	XML query/traversal language adhering to the XPath specification set forth by the W3C.

Index