caGrid 1.0 User's Guide

Center for Bioinformatics

December 15, 2006

caGrid Development and Management Teams		
Scott Oster (Lead Architect)[1]	Ian Foster[2]	Avinash Shanbhag[9]
Stephen Langella[1]	Patrick McConnell[3]	
Shannon Hastings[1]	David Wellborn[4]	
David Ervin[1]	Val Bragg[4]	
Tahsin Kurc[1]	Vinay Kumar[5]	
Joel Saltz[1]	Joshua Phillips[5]	
Ravi Madduri[2]	Ram Chilukuri[5]	
Jarek Gawor[2]	Srini Akkala[5]	
Frank Siebenlist[2]	Manav Kher[6]	
Mike Wilde[2]	Wendy Erickson-Hirons[7]	
Raj Kettimuthu[2]	Arumani Manisundaram[8]	
Bill Allcock[2]	George Komatsoulis[9]	
[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]National Cancer Institute Center for Bioinformatics (NCICB)

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
GenePattern – Project – Broad Institute
geWorkbench – Columbia University
caBiocondutor – Project – Fred Hutchinson Cancer Research Center
Terpsys – Systems Team - National Cancer Institute Center for Bioinformatics (NCICB)

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 1

Purpose 1
Release Schedule 1
Audience 1
Getting Help 1
Using This Guide 1
Document Text Conventions 2

Chapter 2 Overview of caGrid User Roles 3

Overview 3
Relevant Documents 3
User Role Definitions 3
Service Developer 3
Client Application Developer 4
Service Administrator 5

Chapter 3 Developing Analytical and Data Services 7

Overview 7
External Middleware Systems and Tools 8
Introduce Graphical Development Environment 9
Discovery Tools: Support for Strongly-Typed Service Methods 10
Service Creation 10
General 10
caBIG 11
Service Modification 12
Types 13
Operations 14
Resource Properties 15
Service Properties 15
Service Level Security 16
Service Contexts 17
Deployment 19
Using the Introduce Client 19
Creating a Basic Analytical Service 20
Creating a Grid Service Using a Data Extension 34
Creating a Data Service Using the caCORE SDK 41

Chapter 4 Developing Client Applications **49**

Overview 49

caGrid Client APIs 50

Secure Communication *50*

What is an EPR? *51*

Obtaining an EPR for a Service *51*

Inspecting a Service's Metadata *53*

Invoking Operations on a Service *54*

Client Application Case Study: caTRIP 55

caTRIP Discovery Example *55*

caTRIP Metadata Example *56*

caTRIP Invocation Example *56*

Chapter 5 Security Administration **59**

Overview 59

GAARDS Administration User Interface 61

Grid Trust Fabric 62

GTS and the Globus Toolkit *64*

Bootstrapping the Trust Fabric *66*

GTS Software Prerequisites *67*

Building the GTS *67*

Configuring the GTS *67*

Deploying the GTS *68*

Managing Trust Fabric Administrators *69*

Managing Trust Levels *71*

Managing Certificate Authorities *74*

Managing a Federated Grid Trust Fabric *79*

Syncing With the Trust Fabric *83*

Grid User Management 86

Dorian Software Prerequisites *89*

Building Dorian *90*

Configuring Dorian *90*

Deploying Dorian *95*

Setting the Default Administrator Account *95*

Logging onto the Grid *95*

Managing Grid Proxies *96*

Managing Trusted Identity Providers *97*

Grid User Account Management *101*

Local Dorian Identity Provider *104*

Grid Grouper 108

Grid Grouper Software Prerequisites *109*

Building Grid Grouper *110*

Configuring Grid Grouper *110*

Deploying Grid Grouper *111*

Grid Grouper Administration *111*

Administrating Stems *112*
Administrating Groups *117*
Authentication Management 122
Configuring the Service *122*
Deploying to the Container *123*
Configuring the CSM *123*
Authorization Management 125
JAAS Configuration *125*
ApplicationSecurityConfig.xml *125*
hibernate.cfg.xml *126*
Web Applications Classpath *127*
CSM Administration *130*

Chapter 6 Workflow Services 135

Overview 135
The Business Process Execution Language (BPEL) 135
Creating a Workflow 136
Creating a Simple Workflow Using Example Services 136
Submitting the Workflow *136*
Executing the Workflow *137*
Query the Status of the Workflow *137*
Terminating a Workflow *137*
Pausing a Workflow *137*
Resuming a Paused Workflow *137*

Appendix A References 139

Scientific Publications 139
Technical Manuals/Articles 142
caBIG Material 143
caCORE Material 143

Glossary 145

Index 149

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 1.0 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 1.0 release. It may be updated between releases if errors or omissions are found. The current document refers to the 1.0 version of caGrid, released in December 2006 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 Technical 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 whom this guide is written.
- Chapter 3 provides an overview and examples using the Introduce toolkit for service development.
- Chapter 4 introduces the client applications for caGrid services.
- Chapter 5 describes the caGrid security infrastructure, which provides services and tools for to administer and enforce security policy.
- Chapter 6 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.*
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 page 3
- User Role Definitions on page 3

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 document is listed below.

Relevant Documents

This User 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 1.0 Programmer's Guide	http://gforge.nci.nih.gov/frs/?group_id=25
caGrid 1.0 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 Analytical and Data 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. Analytical Services are 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 this 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 that wish 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 that wish 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 wishing to provide additional capabilities (beyond query) in their service may be interested in the entire section.

Client Application Developer

The counterpart to Service Producers are the Client Application Developers, who are the consumers of the content to which the Service Providers provide 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 4 Developing Client Applications. It is highly recommended that Client Application Developers also peruse the caGrid 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, and also provides 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 Security Administration.

Chapter 3 Developing Analytical and Data Services

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

Topics in this chapter include:

- Overview on this page
- External Middleware Systems and Tools on page 8
- Introduce Graphical Development Environment on page 9
- Discovery Tools: Support for Strongly-Typed Service Methods on page 10
- Service Creation on page 10
- Service Modification on page 12
- Deployment on page 19
- Using the Introduce Client on page 19
- Creating a Basic Analytical Service on page 20
- Creating a Grid Service Using a Data Extension on page 34
- Creating a Data Service Using the caCORE SDK on page 41

Overview

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

1) *Creation of 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 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 creates strongly-typed service interfaces using well-defined, published schemas, which are registered in a publicly available grid service registry like the Mobius GME, as the type definitions of the input and output parameters of the service methods.

3) *Deployment.* The developer can deploy the service that has been created with Introduce to a Grid service container (e.g., a Globus or Tomcat service container).

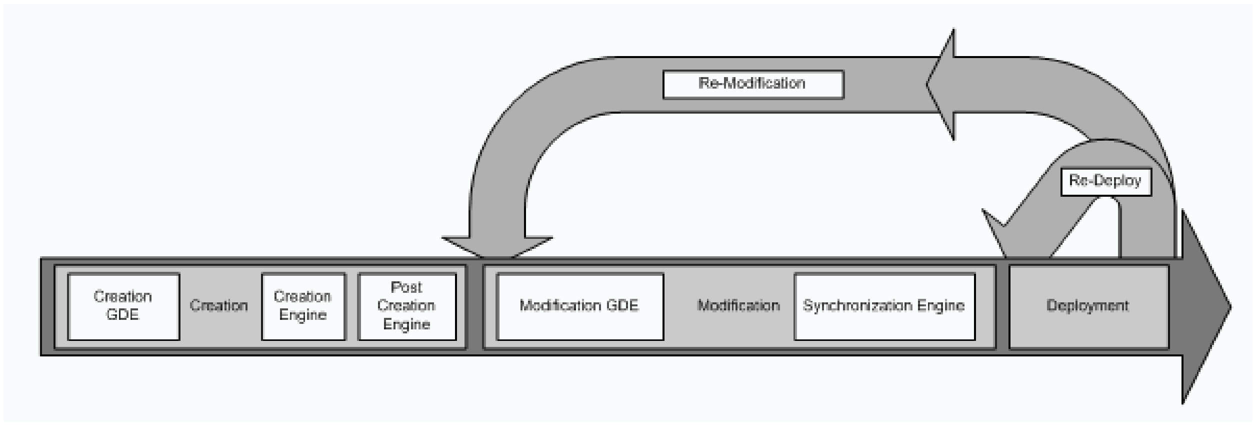


Figure 3-1 Overall service development process

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. In the following sections, the GDE, the Introduce engine, and the extension framework are described in greater detail.

External Middleware Systems and Tools

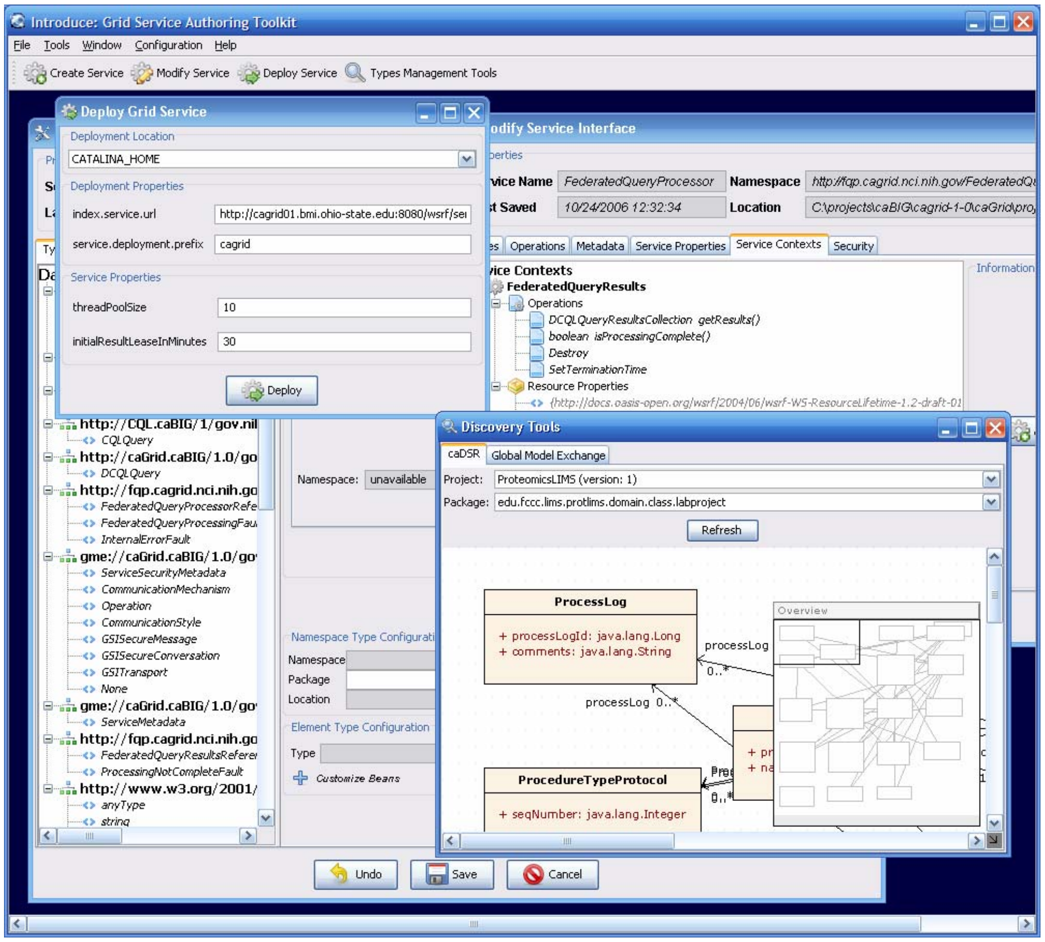
Introduce assumes the availability of two external components to implement its functions: 1) A Grid runtime environment that provides the support for compiling, advertising, and deploying Grid services; 2) A repository of data types that is accessible locally or remotely so that the toolkit can pull 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) is the graphical user interface that can be used to create, modify, and deploy a service (see Figure 2). It is designed to be very simple to use, enable using community excepted data types, and provide easy configuration of service metadata, operations, and security. It also allows customized plug-ins to be added for such things as repositories of data types and for creating custom or common service types.


Introduce: Grid Service Authoring Toolkit
File Tools Window Configuration Help
Create Service Modify Service Deploy Service Types Management Tools
Deploy Grid Service
Deployment Location
Deployment Properties
Service Properties
Deploy
odify Service Interface
Operations Metadata Service Properties Service Contexts Security
Information
Discovery Tools
caDSR Global Model Exchange
Refresh
Overview
Namespace Type Configurati
Element Type Configuration
Customize Beans
Undo Save Cancel

Deployment Location: CATALINA_HOME

index.service.url: http://cagrid01.bmi.ohio-state.edu:8080/wsrf/ser

service.deployment.prefix: cagrid

threadPoolSize: 10

initialResultLeaseInMinutes: 30

vice Name: FederatedQueryProcessor   Namespace: http://fqp.cagrid.nci.nih.gov/FederatedQ

t Saved: 10/24/2006 12:32:34   Location: C:\projects\caBIG\cagrid-1-0\caGrid\proj

vice Contexts
- FederatedQueryResults
  - Operations
    - DCQLQueryResultsCollection getResults()
    - boolean isProcessingComplete()
    - Destroy
    - SetTerminationTime
  - Resource Properties
    - {http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01

- http://CQL.caBIG/1/gov.nil
  - CQLQuery
- http://caGrid.caBIG/1.0/go
  - DCQLQuery
- http://fqp.cagrid.nci.nih.go
  - FederatedQueryProcessorRefe
  - FederatedQueryProcessingFau
  - InternalErrorFault
- gme://caGrid.caBIG/1.0/go
  - ServiceSecurityMetadata
  - CommunicationMechanism
  - Operation
  - CommunicationStyle
  - GSISecureMessage
  - GSISecureConversation
  - GSITransport
  - None
- gme://caGrid.caBIG/1.0/go
  - ServiceMetadata
- http://fqp.cagrid.nci.nih.go
  - FederatedQueryResultsReferer
  - ProcessingNotCompleteFault
- http://www.w3.org/2001/
  - anyType
  - string

Namespace: unavailable

Project: ProteomicsLIMS (version: 1)

Package: edu.fccc.lims.protlims.domain.class.labproject

ProcessLog
+ processLogId: java.lang.Long
+ comments: java.lang.String

ProcedureTypeProtocol
+ seqNumber: java.lang.Integer

processLog 0..*

Figure 3-2 *Introduce Graphical Development Environment (GDE)*

The interface contains several screens and options for the service developer to:

- create a new service
- modify an existing service
- discover and use published data types in order to create strongly-typed service methods

- configure service metadata and deploy the service.

Discovery Tools: Support for Strongly-Typed Service Methods

Using the Introduce GDE, developers can obtain the types that they want to use for the service 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, enables 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 is retrieved, written into the schema area 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 that can be used to locate local schemas. In the caGrid release of Introduce, a caDSR discovery tool plugin is also provided. This tool will be the main way for caGrid users to view and download data types to be used for creating services. The GME plug-in enables developers to browse, upload, and download schemas published in a GME. These schemas represent the valid data types that 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 attempts 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 through the Introduce toolkit. The GME plug-in browser window enables browsing through all the GME published types by namespace and schema name. It presents a quick view of the schema and the option to download the schema bundle. The schema bundle contains the schema and all other schemas that are referenced by that schema.

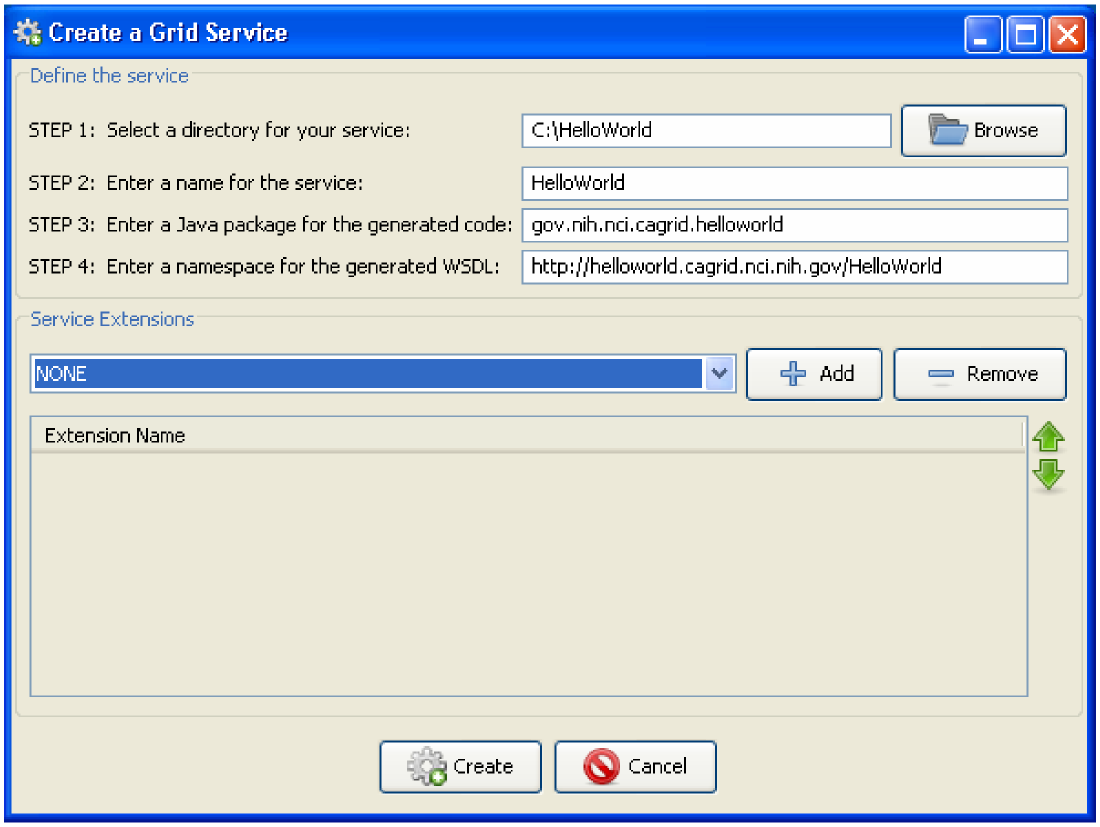
In the caGrid Introduce installation, a discovery plug-in for using caDSR is provided. This plug-in enables the use of types from the caDSR when describing and building grid services. For more information on how the caDSR and the GME work together to provide grid usable data types, please refer to the caGrid Metadata Design document.

Service Creation

General

The service creation component, shown in Figure 3-3, enables a 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 location where the grid service is generated.
- **Service Name:** the name that will be used to generate the service. The service name must be a valid java identifier.
- **Package Name:** the base package to be used when generating the grid service source code.
- **Namespace:** the namespace to be used when defining the WSDL of the service.



STEP 1: Select a directory for your service: C:\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

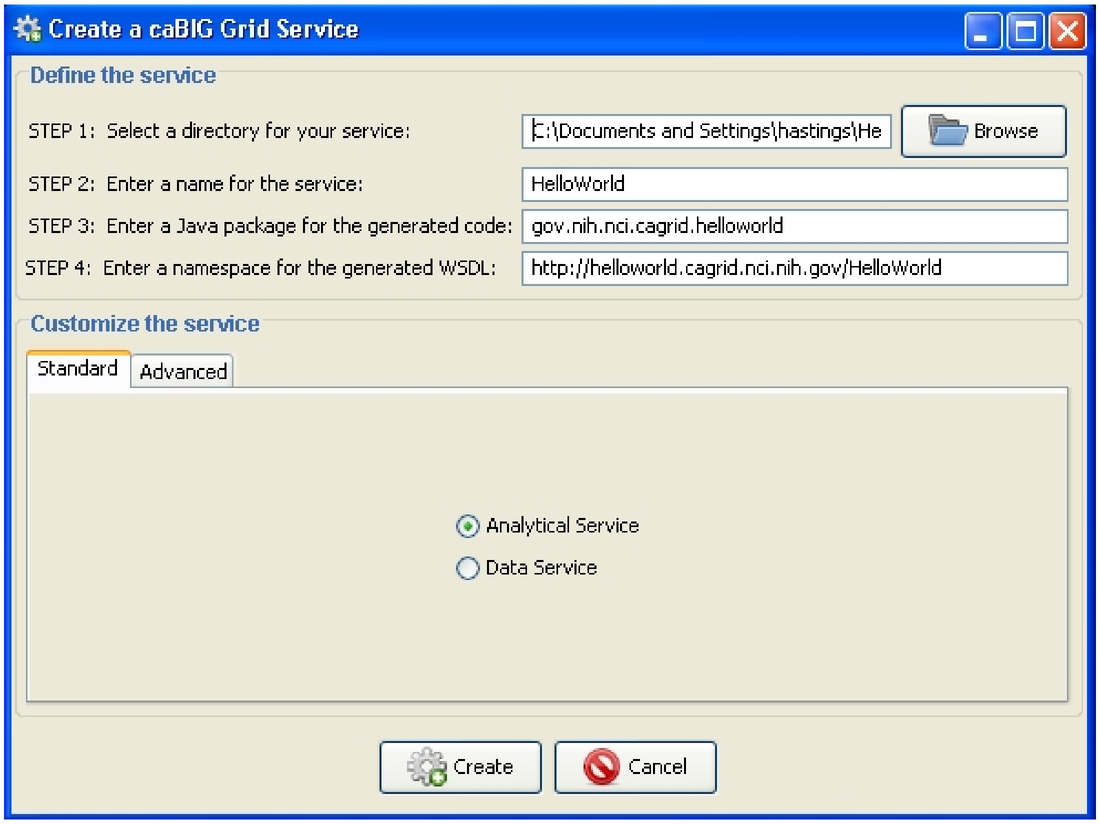
NONE

Figure 3-3 Introduce GDE Service Creation Component

The developer also has the ability to add service extensions. A service extension is an Introduce plug-in that is 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, the user selects the create button. The Introduce creation engine generates the service, and 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 (*Figure 3-4*).



STEP 1: Select a directory for your service: C:\Documents and Settings\hastings\He

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

[✓ Analytical Service] Data Service

Figure 3-4 Introduce caGrid Service Creation Component

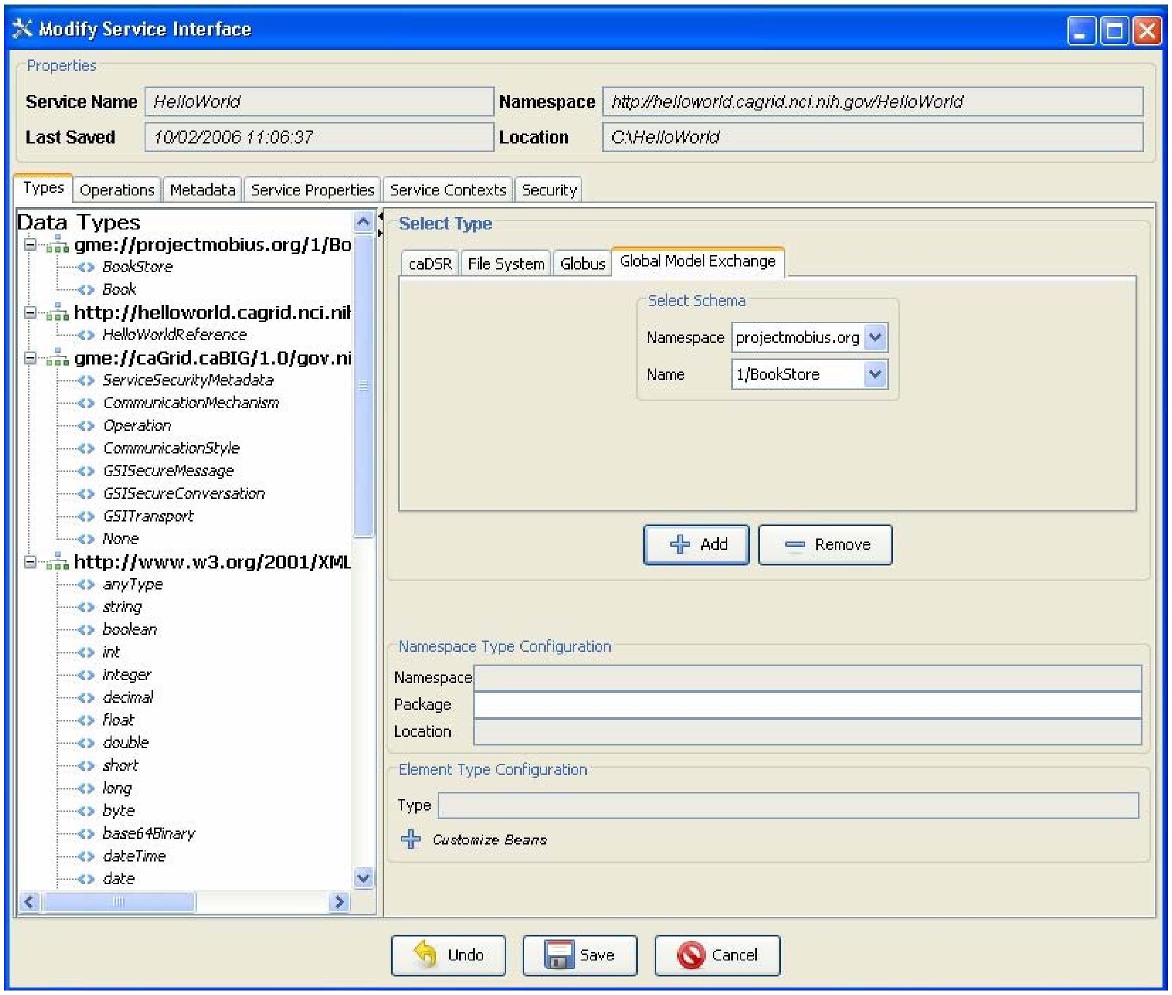
Service Modification

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 in the grid service have been selected and added to the service. Next, the developer 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 how each of the components of the modification viewer can be used to modify the grid service. By selecting the "Modify Service" button on the main menu, a prompt displays to choose the service to be modified. Once the desired directory containing the service to be modified is selected, the modification viewer component is launched. The modification viewer has six main areas where modifications can occur on the service:

- Types
- Operations
- Metadata
- Service Contexts
- Security
- Service Properties

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 types 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).



Service Name: HelloWorld
Namespace: http://helloworld.cagrid.nci.nih.gov/HelloWorld
Last Saved: 10/02/2006 11:06:37
Location: C:\HelloWorld

Namespace: projectmobius.org
Name: 1/BookStore

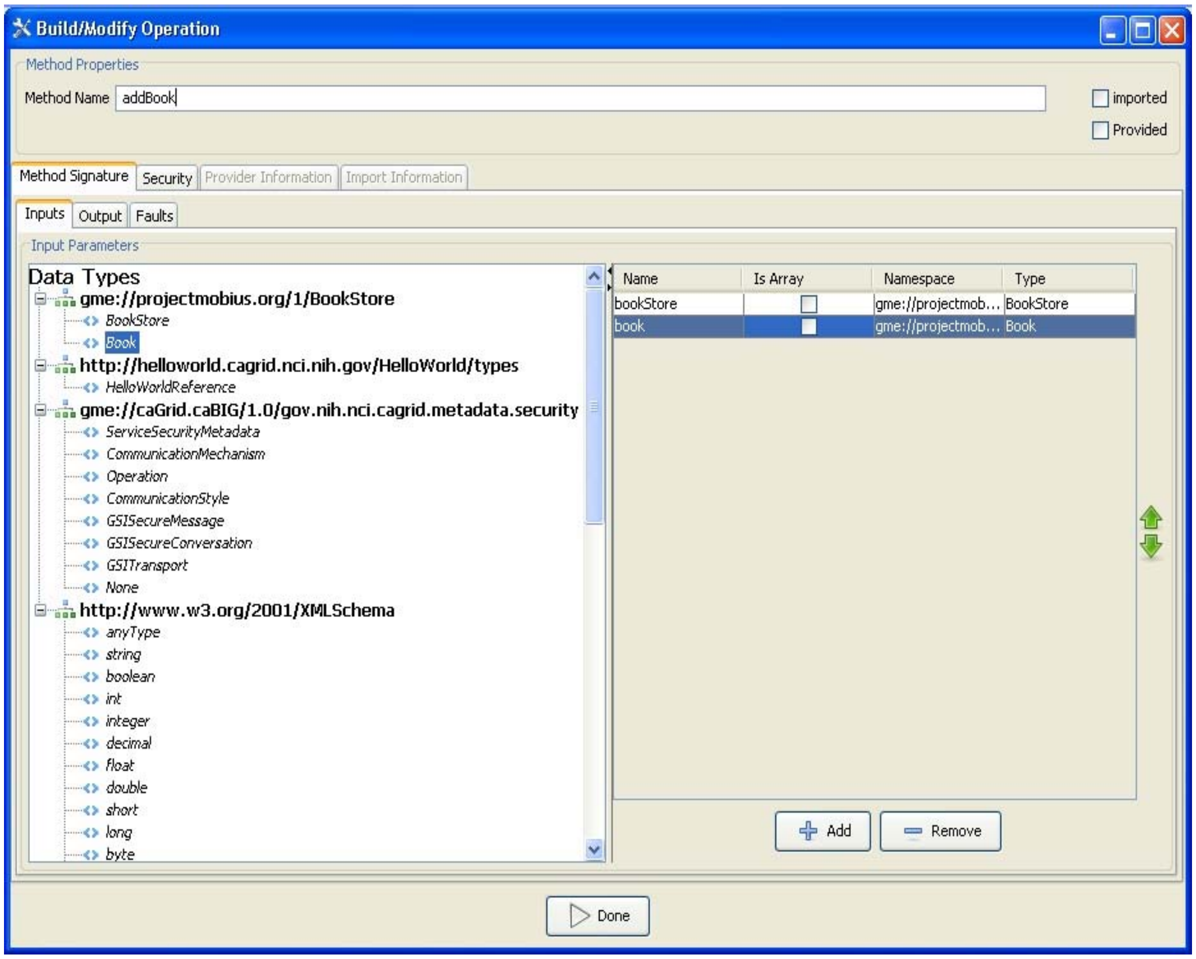
Figure 3-5 Types tab in the Modify Services Interface

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 ways to locate data types and bring them into the service. Currently there are three main discovery mechanisms (GME, Globus, and File System) that are included with Introduce. However, this is extensible via the Discovery Extension described in the Creating a Grid Service Using a Data Extension starting on page 34.

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 are mapped into there 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. If, for example, a set of objects already exist for this particular data type, then a user can decide to provide their own classes and serialization/deserialization factories.

Operations

Using the Operations tab of the GDE Service Modification interface, a developer can add, remove, or modify operations on the service (Figure 3-6).



Method Name: addBook

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

Figure 3-6 Operations tab of the GDE Service Modification interface

For each operation, the developer needs to 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 that 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 name of the parameter is defaulted. This name can be edited by the developer by selecting the cell in the name column and editing the text. There are two ways to add faults, either choose a type from the types tree, which extends WSRF BaseFaultType, or create a new fault, which tells Introduce to create a new fault type to extend the BaseFaultType.

The implementation of a described operation may already exist in another class that is provided by a jar file. You can tell Introduce not to stub this methods server side implementation but instead call this provided method implementation directly in the class provided. In order to use this functionality the “Provided” checkbox must be selected and the Class name attribute must be filled out in the “Provider” tab. The jar file that contains the provided Class that implements this operation must also be copied into the lib directory of the service. This ensures that the operation is located at the time the operation is called on the service.

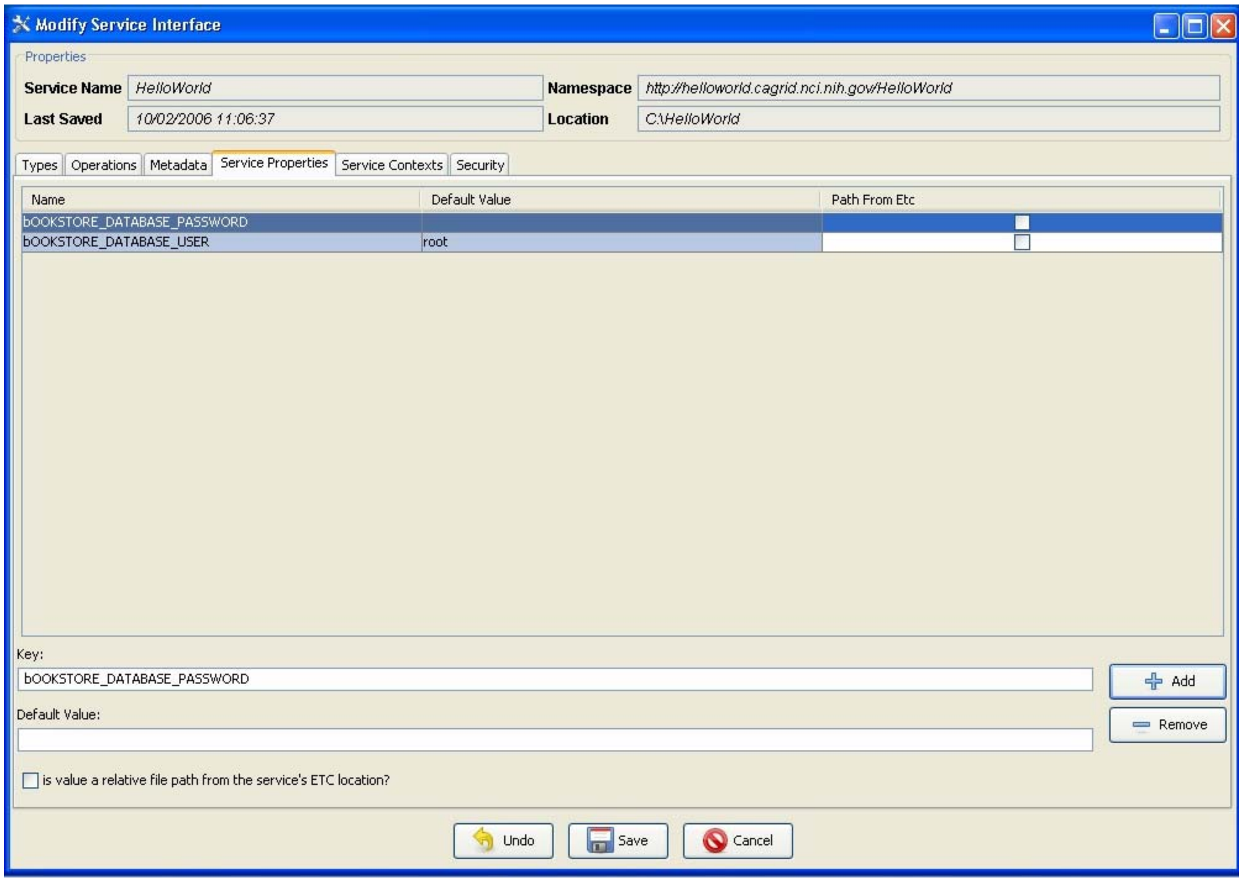
Operations can also be *imported* from other services. Importing an operation enables the service to implement the exact same operation signature. This enables the service to have an operation that has the exact same WSDL signature of the operation being imported. This enables either client to call this operation on either service. Importing can be done in two ways: (1) from an Introduce generated service, or (2) from a WSDL file. For case 1, importing from an Introduce service, the developer browses and selects the Introduce generated service that contains the operation to be imported. Once the Introduce service is selected, a list of services that contain this method are available to select. Select the service from which you want to import the operation. The methods signature is imported and the developer is prompted to copy the WSDL and XSD files needed to import the method into the *schema<servicename>* directory of the service. For case 2, if a method is described in another WSDL but the developer wants to implement this exact method from this WSDL, the developer must have the WSDL and corresponding XSDs in the *schema/<servicename>* directory of the service. Then the developer can browse those WSDL files and select the port type they wish to import the operation from. Importing a method across services assures not only that each service has completely 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.

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 that 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.

Service Properties

Service properties are key value pairs that are set at deployment time and are available to the server side implementation of the service. This enables passing in configuration variables to the server side of the service at deployment. These key value pair properties are declared in the Service Properties tab of the GDE Service Modification interface (Figure 3-7), and a default value can be given there as well. The properties are confirmed and/or can be changed at deployment time. The variables can then be accessed inside the user’s implementation of the operations.



Service Name: HelloWorld
Namespace: http://helloworld.cagrid.nci.nih.gov/HelloWorld
Last Saved: 10/02/2006 11:06:37
Location: C:\HelloWorld

Name	Default Value	Path From Etc
bOOKSTORE_DATABASE_PASSWORD		☐
bOOKSTORE_DATABASE_USER	root	☐

Key: bOOKSTORE_DATABASE_PASSWORD

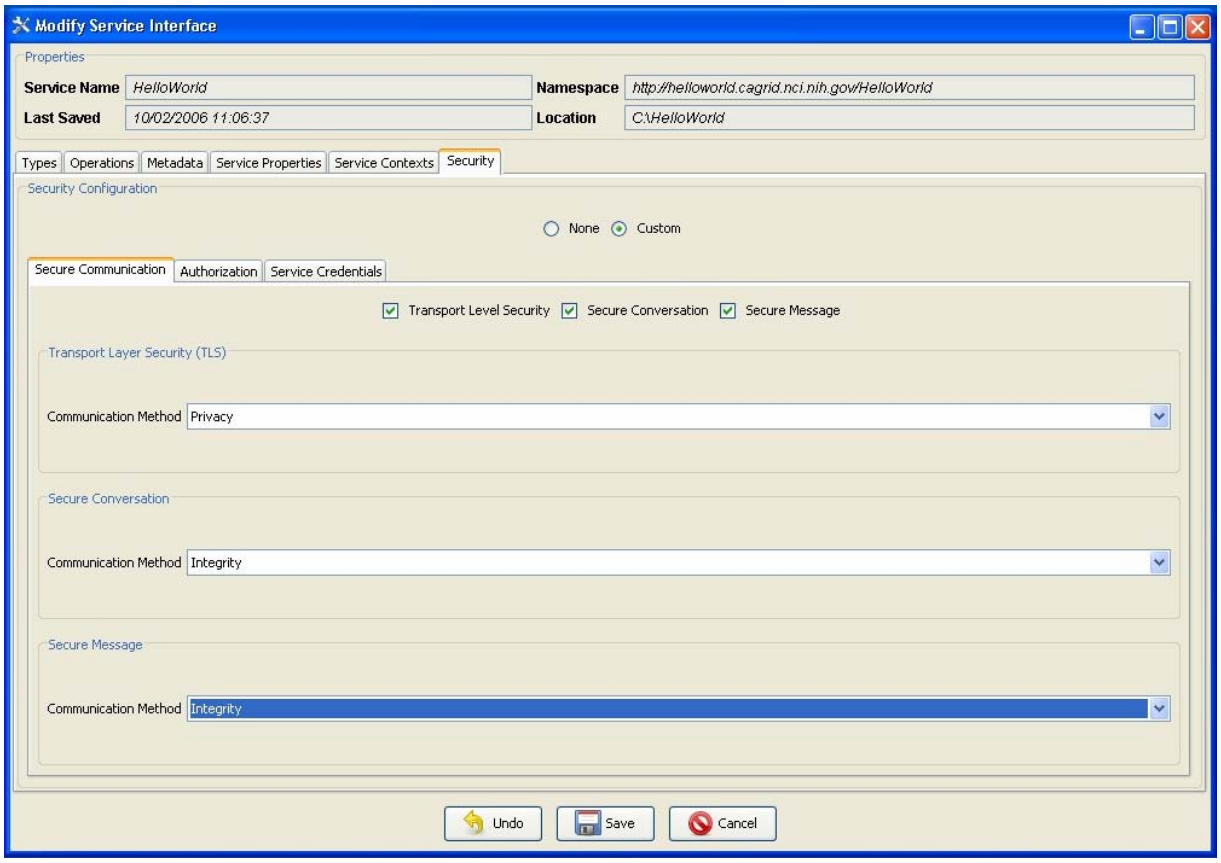
Default Value:

☐ is value a relative file path from the service's ETC location?

Figure 3-7 Service Properties tab of the GDE Service Modification interface

Service Level Security

Service level security configuration can be set at this time as well via the Security tab of the GDE Service Modification interface (Figure 3-8). The service level security can be superseded by method level security. For example, if a service does not have any service level security constraints but a particular method needs to be secured, the secure method level configuration takes precedence over the service level security configuration.


Modify Service Interface

Properties

Service Name	HelloWorld	Namespace	http://helloworld.cagrid.nci.nih.gov/HelloWorld
Last Saved	10/02/2006 11:06:37	Location	C:\HelloWorld

Security Configuration: None [✓ Custom]

Secure Communication: Transport Level Security ✓, Secure Conversation ✓, Secure Message ✓

Transport Layer Security (TLS) — Communication Method: Privacy

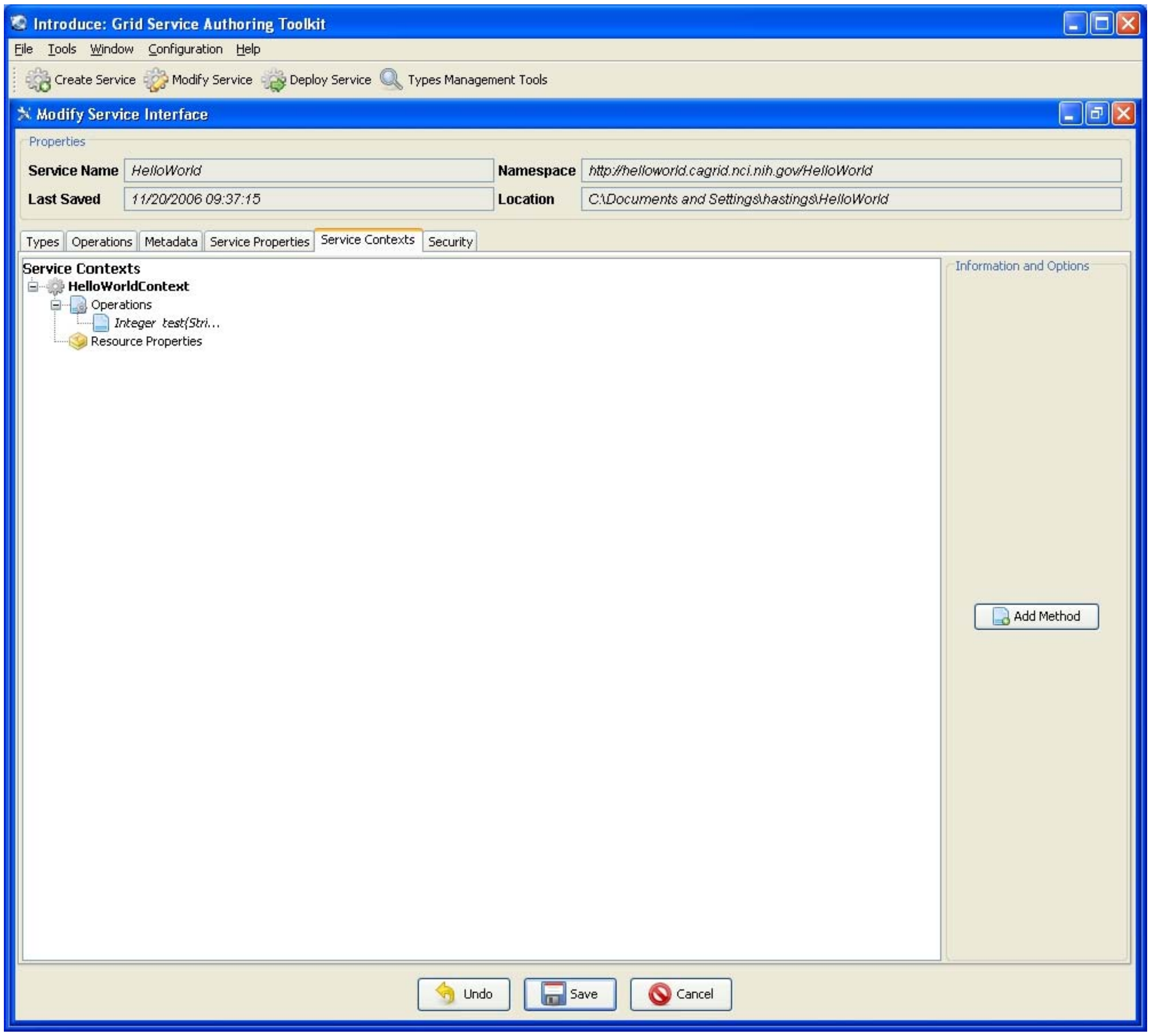
Secure Conversation — Communication Method: Integrity

Secure Message — Communication Method: Integrity

Figure 3-8 Security tab of the GDE Service Modification interface

Service Contexts

An advanced feature that can be enabled at modification time is the addition or removal of service contexts. Contexts can be added via the Service Contexts tab of the GDE Service Modification interface (Figure 3-9). Service contexts define additional conceptual contexts of operations needed to support the desired service functionality.



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

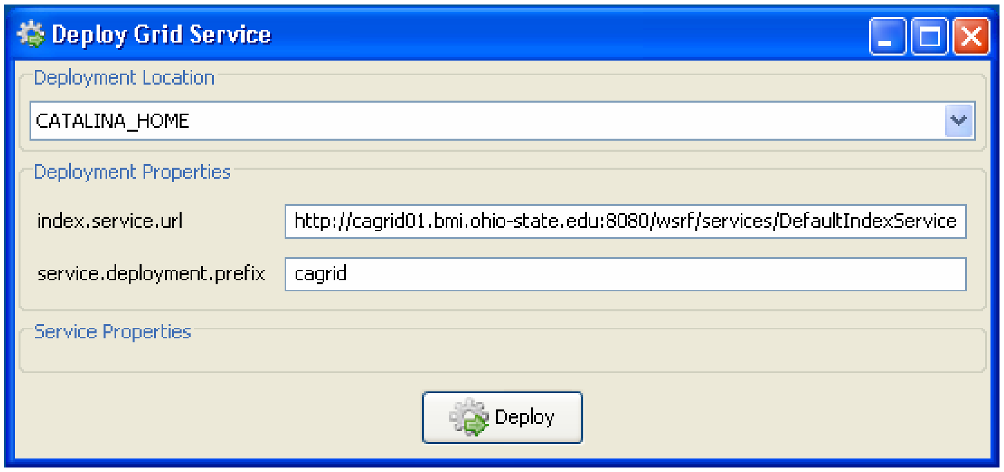
Service Contexts
HelloWorldContext
Operations
Integer test(Stri...
Resource Properties

Figure 3-9 Service Contexts tab of the GDE Service Modification interface

This is enabled by using WSRF capabilities of the Globus Toolkit. As an example, if an operation on the main service enables the user to query a database, 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 its worth noting how they are realized, and 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.

Deployment

The *Deployment* component of the GDE (*Figure 3-10*) allows the service developer to deploy the implemented grid service, which has been created with Introduce, to a Grid service container. 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 window allows the service deployer to populate 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

index.service.url: http://cagrid01.bmi.ohio-state.edu:8080/wsrf/services/DefaultIndexService

service.deployment.prefix: cagrid

Figure 3-10 Introduce GDE service deployment component

Using the Introduce Client

Introduce generates a client API for the service that is described within the graphical editing environment (see Operations on page 14). This client API can be used in order to leverage this type of service from another application or service. The API contains four constructors that can use used, each of which is different depending on whether one has a handle or just an address and the need for security to be used.

Once a client handle is constructed, each of the operations that were created in the service is available as operations to this newly constructed client instance. Figure 3-11 contains an example snippet of code that creates a new client handle to a service called “HelloWorld”. Figure 3-12 calls the “echo” operation.

```
/**
 * 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 3-11 New client handle to service “HelloWorld”

```
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 3-12 Call to “echo” operation

Creating a Basic Analytical Service

This section provides specific steps showing how to use the Introduce Grid Service Authoring Toolkit to build a basic analytical grid service. The example used is that of a book store. The latter steps illustrate more advanced tools for configuring metadata, setting grid level and method level security, and setting deployment options.

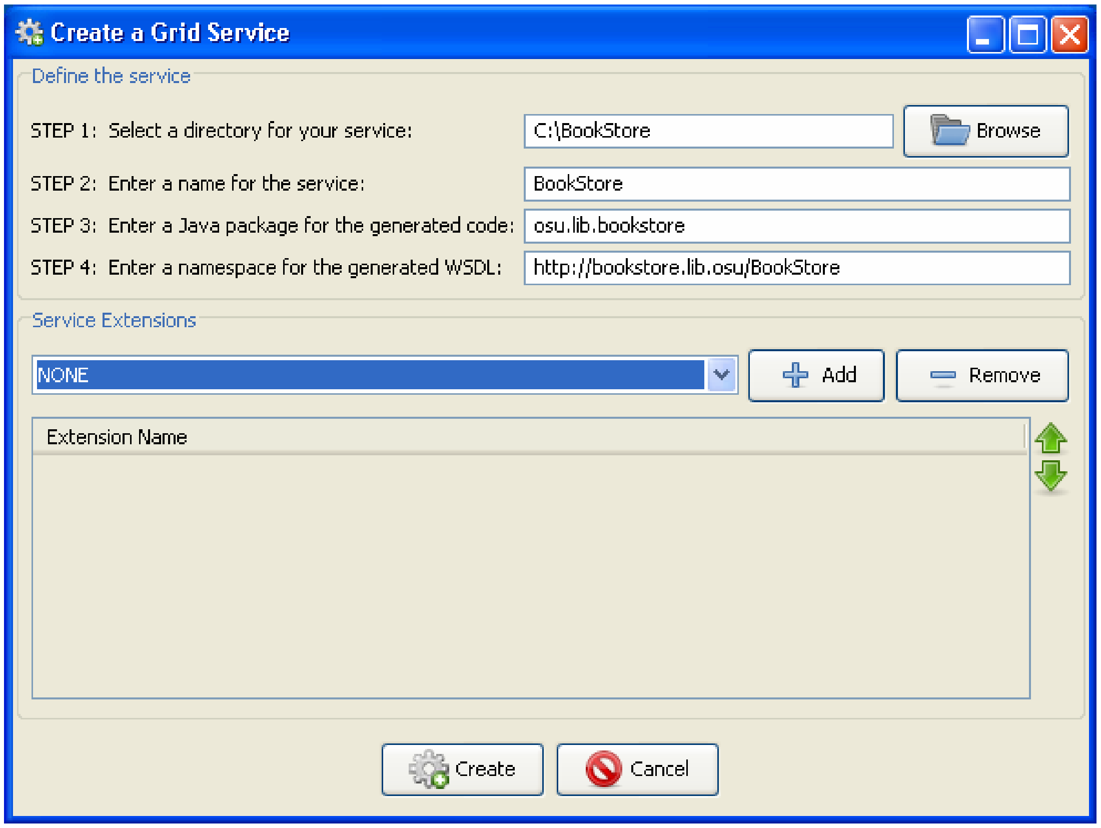
The following steps are outlined:

- Example One: Create a Basic Service
- Example Two: Create or Locate Data Types
- Example Three: Add Methods to the Service
- Example Four: Implement the Methods
- Example Five: Deploy the Service
- Example Six: Implement a Simple Client
- Example Seven: Start the Globus Container and Run the Client
- Example Eight: Populating caGrid Service Metadata

Example One: Create a Base Grid Service

This example provides instructions to create a base grid service, add data types, add a method, and implement that method. To create a base service, use the following steps:

1. To launch the Introduce Grid Service Authoring Toolkit, invoke *ant introduce* from the top level of the Introduce release.
2. Select **Create Service** from the toolbar or select **Tools->Create Service** from the toolbar. The Create a Grid Service window displays (Figure 3-13).
   Note: If you are a caGrid user, use the create caGrid service instead.
3. Select a directory to place the service. This example uses “C:\BookStore”.
4. Enter a name for the service. This example uses “BookStore”.
5. Enter a Java package for the generated code. This example uses “osu.lib.bookstore”.
6. Enter a namespace for the generated WSDL. This example uses “http://bookstore.osu.lib/BookStore”.
7. Click the **Create** button at the bottom of the window. It may take several minutes for Introduce to create the service.



STEP 1: Select a directory for your service: C:\BookStore

STEP 2: Enter a name for the service: BookStore

STEP 3: Enter a Java package for the generated code: osu.lib.bookstore

STEP 4: Enter a namespace for the generated WSDL: http://bookstore.lib.osu/BookStore

NONE

Figure 3-13 Create a base grid service

Example Two: Create or Locate Data Types

This example provides instructions to create or locate data types to add to an existing base grid service, which was created in the previous example Creating a Basic Analytical Service

In Introduce, new data types can be obtained in three ways:

- create a data type and upload it from the file system
- download a data type from the GME
- import a data type from Globus, or in the case of caGrid users, there is a caDSR plug-in.

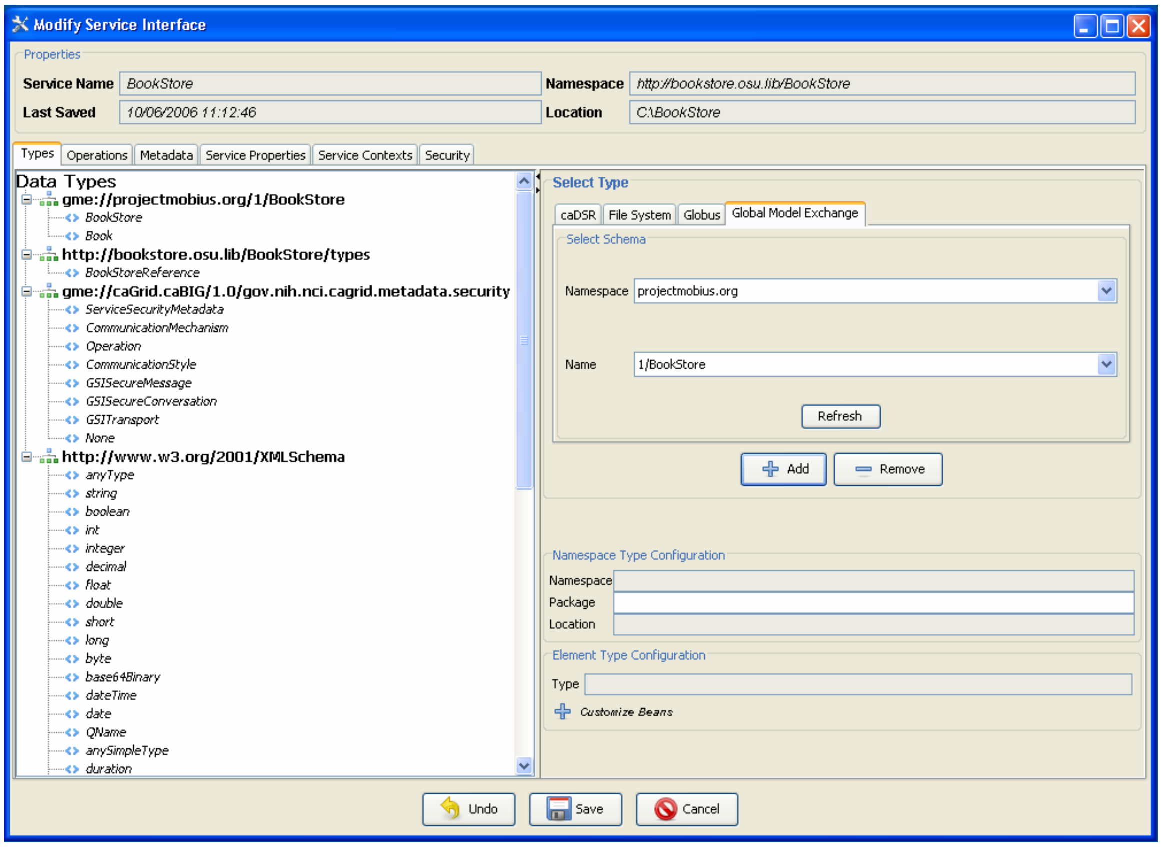
In this example, data types are downloaded using the file system browser. Use the following steps to upload a data type to the GME and obtain it for the service being built with Introduce.

1. Download the schema file from the following location and save it to an accessible location on the file system: http://bmi.osu.edu/~hastings/introduce/tutorial/bookstore.xsd The schema is now accessible for Introduce. The following steps outline how to import the schema via the File System Discovery Tool.

2. **Note:** The Modify Service window automatically after grid service is initially created. If this is the case, go to step 3.

   Select **Modify Service** from the toolbar or select **Tools->Modify Service** to open the **Select Directory** window. Select the root directory where the grid service for the book store example is located. The directory for this tutorial is *C:\BookStore*. The Modify Service Interface window opens.

3. This step imports the schema *1_BookStore* into the service. From the Modify Service Interface window, verify the **Types** tab is active and then browse to the **Select Type** section located on the right side of the window. In the Select Type window, select the **File System** tab. Browse to the schema file that you saved in the previous step. Click the **Add** button (Figure 3-14).
   Note: The types *BookStore* and *Book* should display in the Data Types section on the left side of the window.



Modify Service Interface

Properties

Service Name: BookStore
Namespace: http://bookstore.osu.lib/BookStore
Last Saved: 10/06/2006 11:12:46
Location: C:\BookStore

Data Types
- gme://projectmobius.org/1/BookStore
  - BookStore
  - Book
- http://bookstore.osu.lib/BookStore/types
  - BookStoreReference
- 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
  - base64Binary
  - dateTime
  - date
  - QName
  - anySimpleType
  - duration

Select Type

Select Schema

Namespace: projectmobius.org
Name: 1/BookStore

Namespace Type Configuration

Namespace:
Package:
Location:

Element Type Configuration

Type:

Figure 3-14 Modify Service Interface

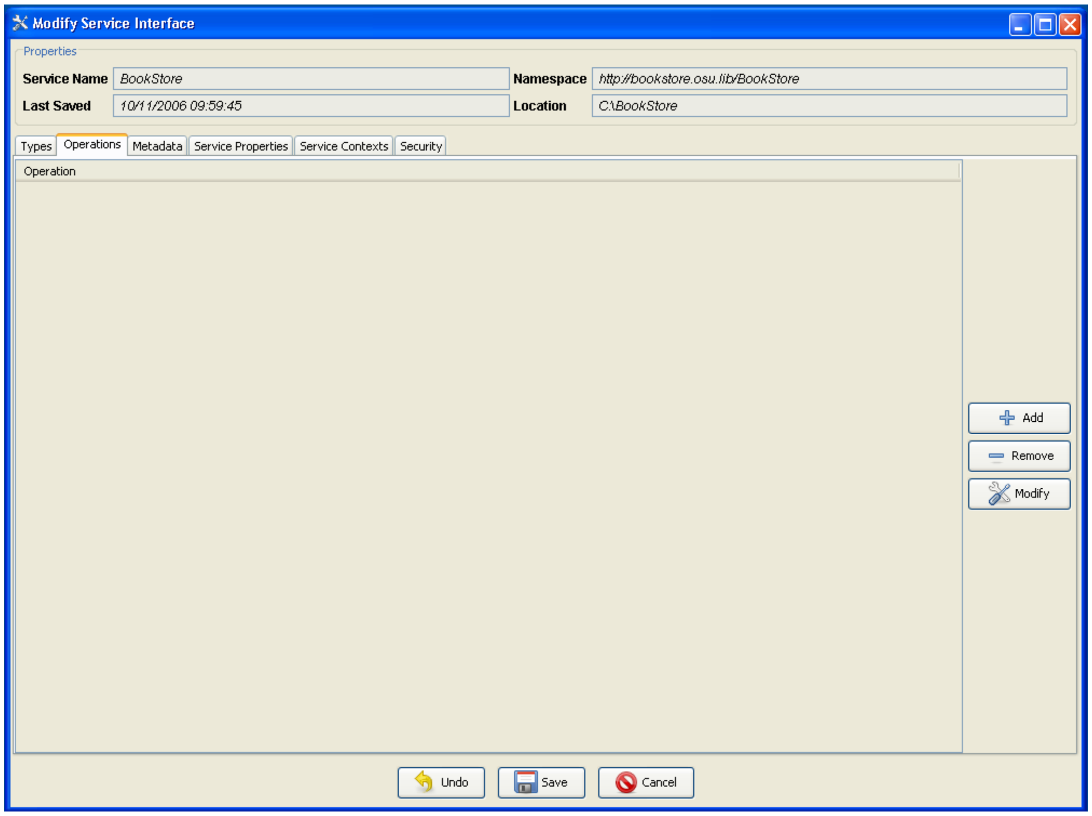
4. Click the **Save** button at the bottom of the Modify Service Interface window.

Example Three: Add Methods to the Service

This example provides instructions to add to a service. The added methods have an input parameter, an output parameter, and throw exceptions. Use the following steps to add method to the service:

1. Open the *BookStore* service to be modified.
   Note: Refer to steps 2 and 3 on page 23 of Example Two: Create or Locate Data Types for instruction.

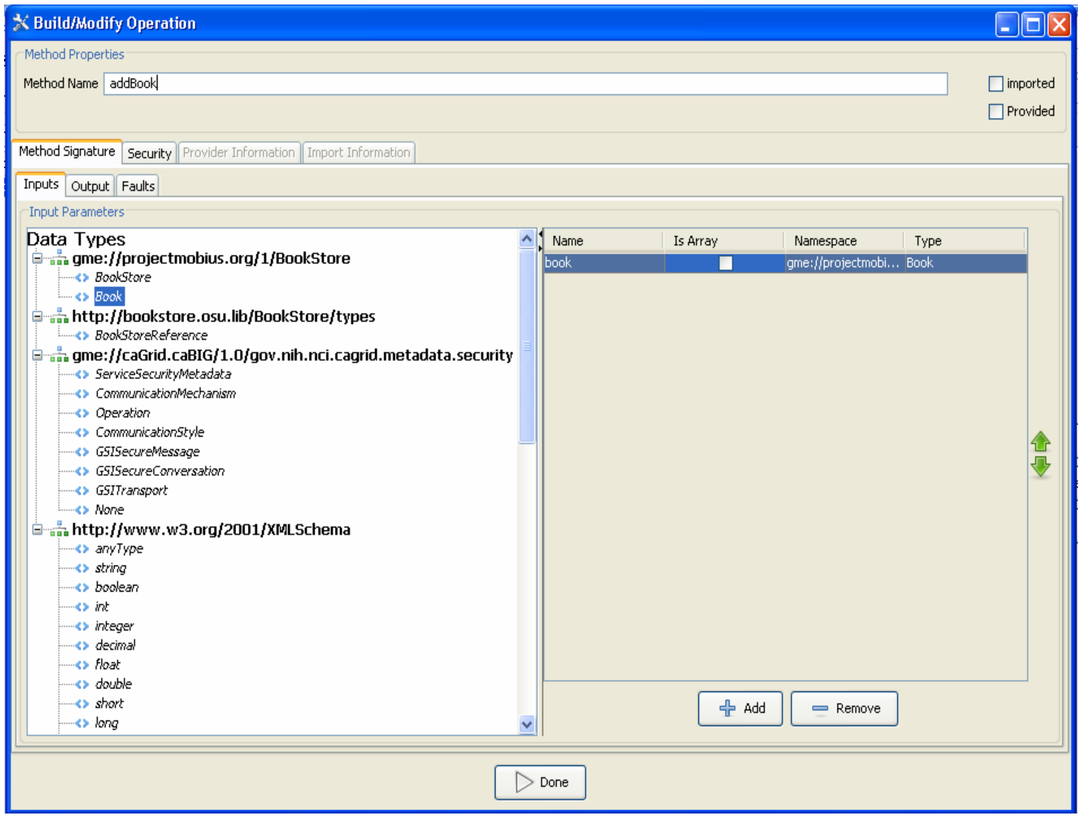
2. Click on the **Operations** tab in the Modify Service Interface window (Figure 3-15 Operations Tab).



Service Name: BookStore
Namespace: http://bookstore.osu.lib/BookStore
Last Saved: 10/11/2006 09:59:45
Location: C:\BookStore

Figure 3-15 Operations Tab

3. On the Operations tab, click **Add** on the right hand side of the window. The **Build/Modify Operations** window opens.
4. In the Build/Modify Operations window, in the Method Name field, enter *addBook* where *newMethod* displays. Select the **Inputs** tab (if it is not already) and select **Book** from the Data Types tree. Click **Add** in the lower right hand portion of the window (Figure 3-16).
   Note: The input for the *getBook* method will be of type string.



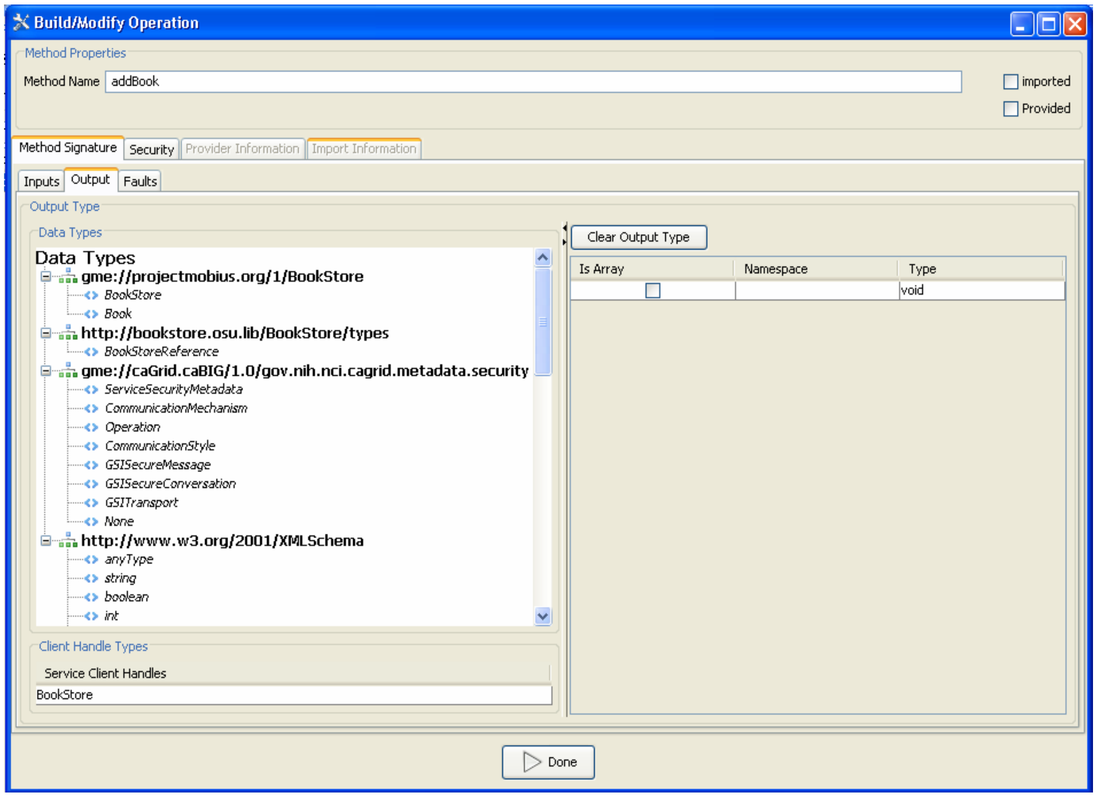
Method Name: addBook

Data Types
- gme://projectmobius.org/1/BookStore
  - BookStore
  - Book
- http://bookstore.osu.lib/BookStore/types
  - BookStoreReference
- 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

Name	Is Array	Namespace	Type
book	☐	gme://projectmobi…	Book

Figure 3-16 Add Input Parameter

5. Select the **Output** tab in the Build/Modify Operation window. By default, the output type is void. For the *addBook* method, leave the output as void (Figure 3-17).
   Note: For the method *getBook*, the output type will be of type Book. Double click the data type to set the output type.



Method Name: addBook

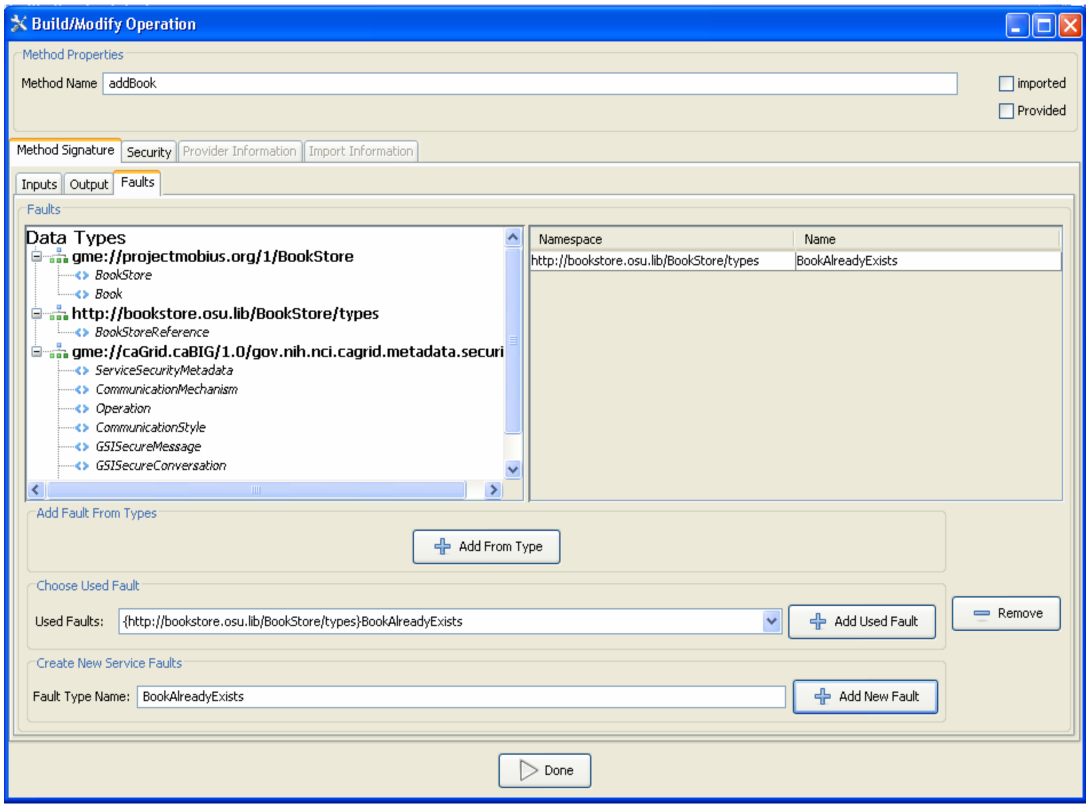
Data Types
- gme://projectmobius.org/1/BookStore
  - BookStore
  - Book
- http://bookstore.osu.lib/BookStore/types
  - BookStoreReference
- gme://caGrid.caBIG/1.0/gov.nih.nci.cagrid.metadata.security
  - ServiceSecurityMetadata
  - CommunicationMechanism
  - Operation
  - CommunicationStyle
  - GSISecureMessage
  - GSISecureConversation
  - GSITransport
  - None
- http://www.w3.org/2001/XMLSchema
  - anyType
  - string
  - boolean
  - int

Is Array	Namespace	Type
☐		void

Service Client Handles: BookStore

Figure 3-17 Add Output Parameter

6. Finally, select the **Faults** tab from the Build/Modify Operations window. At the bottom of the Faults window, enter *BookAlreadyExists* in the **Fault Type Name** field. Click **Add New Fault**. The fault populates values for Namespace and Name in the Faults window (Figure 3-18).
7. Click **Done**. The Modify Service Interface screen displays.



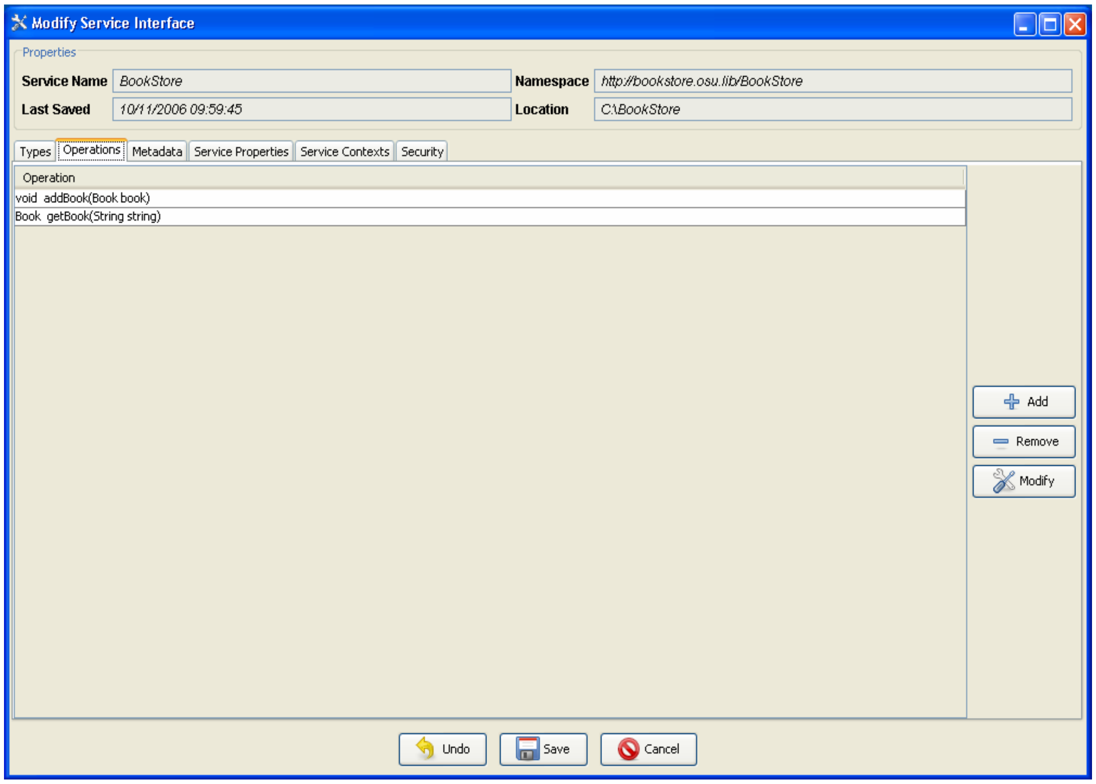
Namespace	Name
http://bookstore.osu.lib/BookStore/types	BookAlreadyExists

Used Faults: {http://bookstore.osu.lib/BookStore/types}BookAlreadyExists

Fault Type Name: BookAlreadyExists

Figure 3-18 Add Fault

8. Use steps 1 through 7 to add the second method *getBook* with string as the input parameter, *Book* as the output parameter, and *BookDoesNotExist* as the fault.
9. Refer to Figure 8 to verify the methods have been added properly. Click **Save** when finished.


Modify Service Interface

Properties

Service Name: BookStore
Namespace: http://bookstore.osu.lib/BookStore
Last Saved: 10/11/2006 09:59:45
Location: C:\BookStore

Operation
void addBook(Book book)
Book getBook(String string)

Figure 3-19 Added Methods

Example Four: Implement the Methods

The one aspect that Introduce does not automatically generate when creating a grid service is the actual implementation for the created methods. This example provides trivial implementations to the methods created in the previous section. *addBook* is a method that takes in type *Book* and returns type *void*. *getBook* is a method that takes in type *String* and returns type *Book*. The implementations here demonstrate the process but have no functional purpose. Use the following steps to provide implementations to created methods.

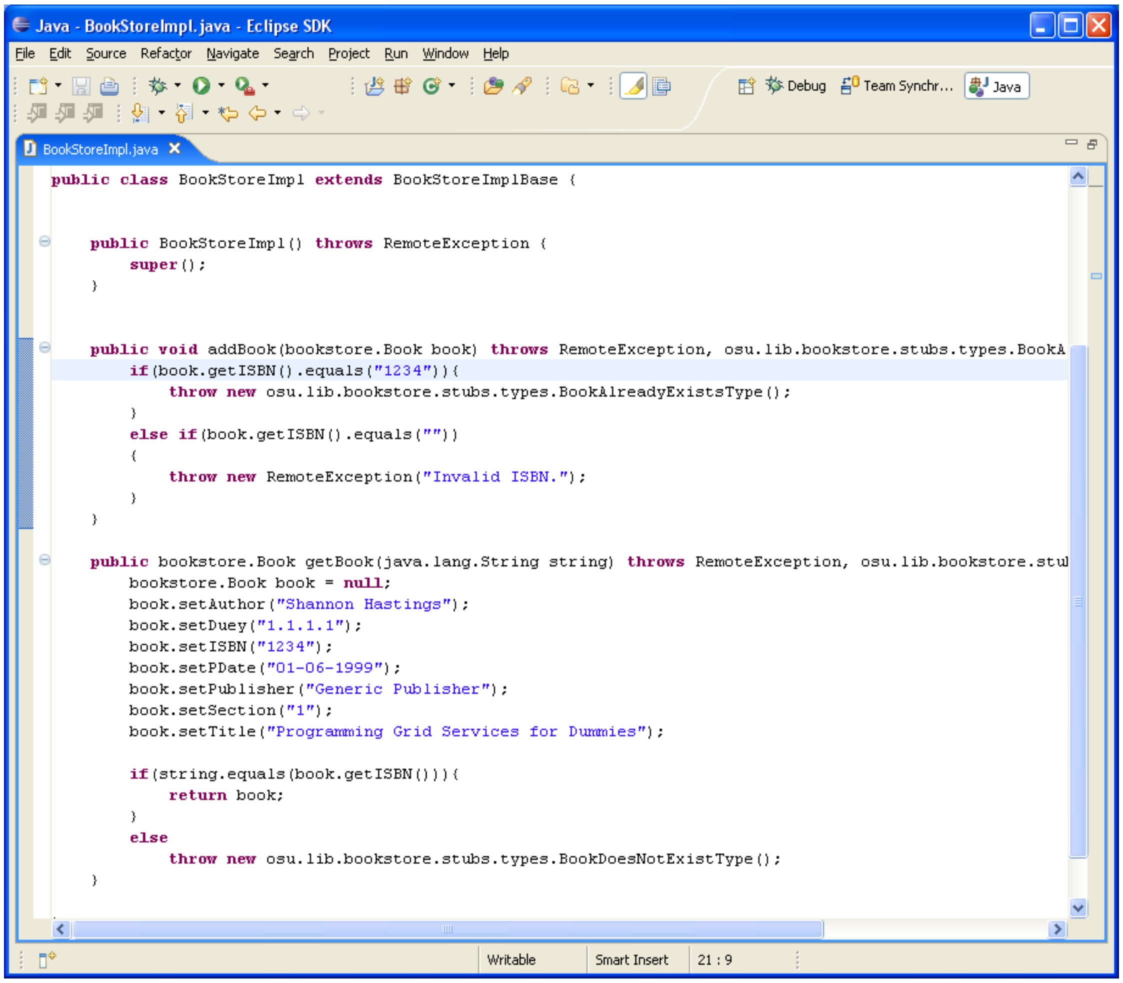
1. Browse to the location of the *Bookstore* service at *C:\BookStore* and locate the folder *C:\BookStore\src\osu\lib\bookstore\service*. This folder contains the file *BookStoreImpl.java*, which was automatically generated by Introduce.
2. The two methods, *addBook* and *getBook*, are located in this file and need to be implemented. Open *BookStoreImpl.java* with a Java editor or text editor. Copy and paste the following code to implement the *addBook* operation:

```
    if(book.getISBN().equals("1234")){

    throw new osu.lib.bookstore.stubs.types.BookAlreadyExistsType();

}

else if(book.getISBN().equals(""))

{

    throw new RemoteException("Invalid ISBN.");

}
```

3. Copy and paste the following code for *getBook*:

```
        bookstore.Book book = null;
    book.setAuthor("Shannon Hastings");
    book.setDuey("1.1.1.1");
    book.setISBN("1234");
    book.setPDate("01-06-1999");
    book.setPublisher("Generic Publisher");
    book.setSection("1");
    book.setTitle("Programming Grid Services for Dummies");

    if(string.equals(book.getISBN())){
        return book;
    }
    else
        throw new osu.lib.bookstore.stubs.types.BookDoesNotExistType();
```

4. Refer to Figure 3-20 to validate the previous steps.


Java - BookStoreImpl.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Team Synchr...
Java
BookStoreImpl.java
Writable
Smart Insert
21 : 9

```
public class BookStoreImpl extends BookStoreImplBase {


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


    public void addBook(bookstore.Book book) throws RemoteException, osu.lib.bookstore.stubs.types.BookA
        if(book.getISBN().equals("1234")){
            throw new osu.lib.bookstore.stubs.types.BookAlreadyExistsType();
        }
        else if(book.getISBN().equals(""))
        {
            throw new RemoteException("Invalid ISBN.");
        }
    }

    public bookstore.Book getBook(java.lang.String string) throws RemoteException, osu.lib.bookstore.stu
        bookstore.Book book = null;
        book.setAuthor("Shannon Hastings");
        book.setDuey("1.1.1.1");
        book.setISBN("1234");
        book.setPDate("01-06-1999");
        book.setPublisher("Generic Publisher");
        book.setSection("1");
        book.setTitle("Programming Grid Services for Dummies");

        if(string.equals(book.getISBN())){
            return book;
        }
        else
            throw new osu.lib.bookstore.stubs.types.BookDoesNotExistType();
    }
```

Figure 3-20 BookStoreImpl.java

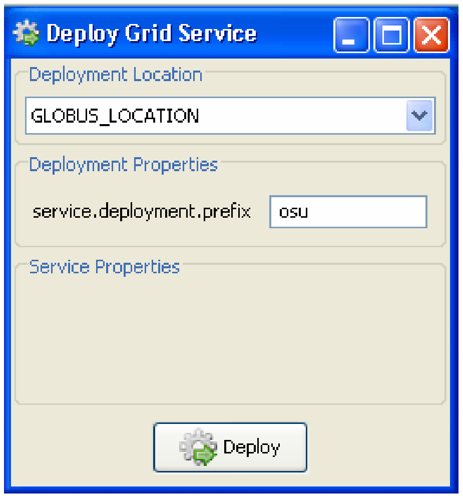
5. Save the changes.

Example Five: Deploy the Service

Deploying a service creates a *Gar* file that is then extracted into a container. Introduce allows a service to be deployed to either Tomcat or Globus. The example provides instruction on deploying to Globus. For more information, refer to the Introduce Technical Guide.

Note: Environmental variables need to be set for Globus (GLOBUS_LOCATION) and Tomcat (CATALINA_HOME) in order to use Introduce to deploy a service.

1. Click **Deploy Service** or select **Tools->Deploy Service** from the Introduce window main menu. The **Select Directory** window opens. Select the directory that contains *BookStore* service (*C:/BookStore* for this example).
2. Click **Open**. The **Deploy Grid Service** window opens.
3. Select GLOBUS_LOCATION under the Deployment Location section.
4. Enter *osu* for the service.deployment.prefix under Deployment Properties (Figure 3-21). Click **Deploy**.



Deployment Location: GLOBUS_LOCATION

Deployment Properties

service.deployment.prefix: osu

Service Properties

Figure 3-21 Deploy Grid Service

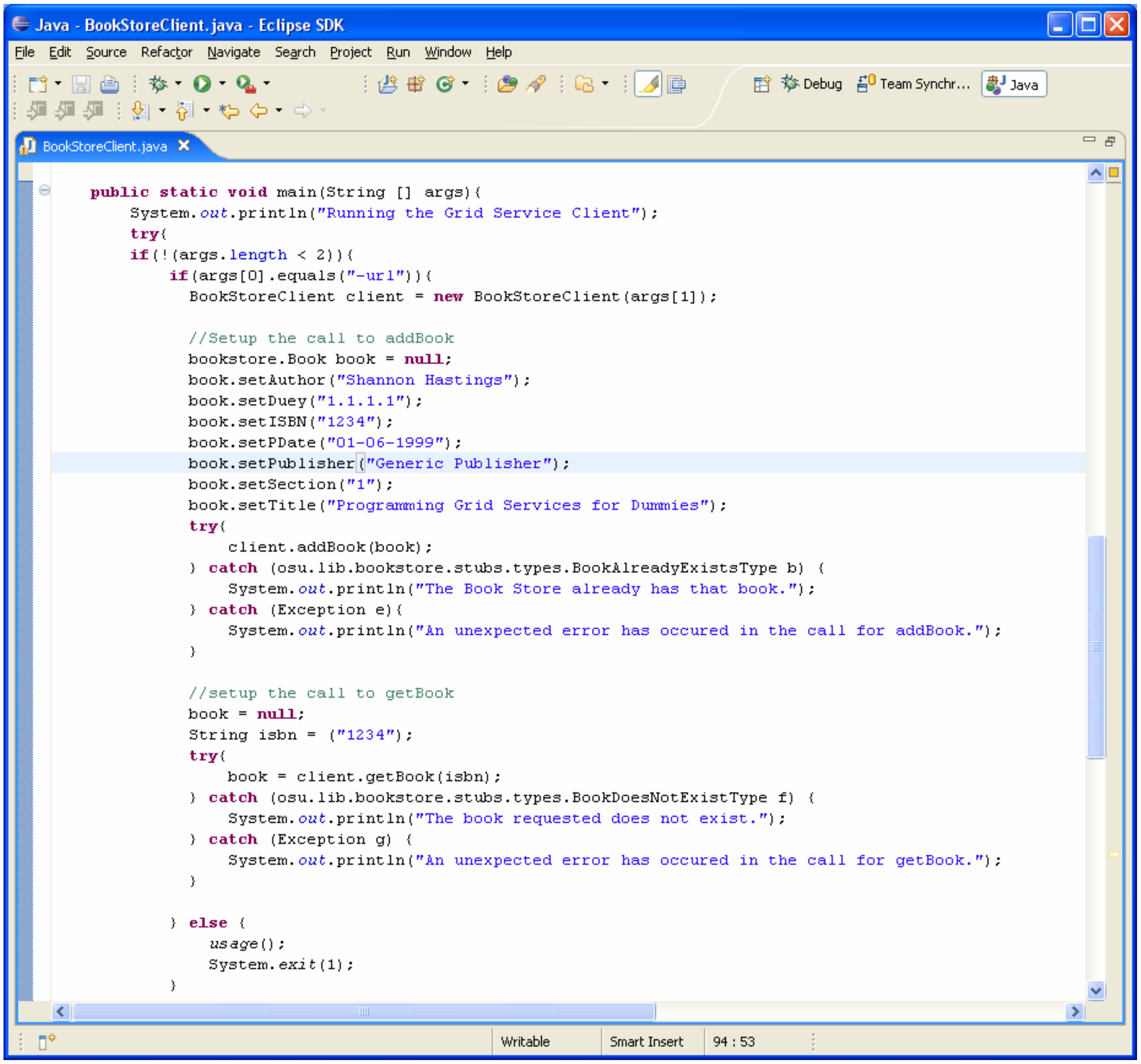
Example Six: Implement a Simple Client

In order to test the service created in this example, a simple client can be created to make remote calls to the deployed service. Introduce creates a template that can easily be filled in to accomplish this task. Use the following steps to fill in the template.

1. From the BookStore directory (*C:/BookStore* for this example), open the following file with either a java editor or text editor:
   C:\BookStore\src\osu\lib\bookstore\client\BookStoreClient.java
2. Copy and paste the following code into the main method of the Bookstore client as shown in Figure 3-22:

```
            //Setup the call to addBook
         bookstore.Book book = null;
         book.setAuthor("Shannon Hastings");
         book.setDuey("1.1.1.1");
      book.setISBN("1234");
      book.setPDate("01-06-1999");
      book.setPublisher("Generic Publisher");
         book.setSection("1");
         book.setTitle("Programming Grid Services for Dummies");
      try{
              client.addBook(book);
            } catch(osu.lib.bookstore.stubs.types.BookAlreadyExistsType b) {
         System.out.println("The Book Store already has that book.");
          } catch (Exception e){
                System.out.println("An unexpected error has occurred in the
            call for addBook.");
```

```
                }

      //setup the call to getBook
         book = null;
      String isbn = ("1234");
      try{
      book = client.getBook(isbn);
      } catch (osu.lib.bookstore.stubs.types.BookDoesNotExistType f) {
      System.out.println("The book requested does not exist.");
      } catch (Exception g) {
      System.out.println("An unexpected error has occurred in the call for
    getBook.");
      }
```


Java - BookStoreClient.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Team Synchr...
Java
BookStoreClient.java
Writable
Smart Insert
94 : 53

```
    public static void main(String [] args){
        System.out.println("Running the Grid Service Client");
        try{
        if(!(args.length < 2)){
            if(args[0].equals("-url")){
              BookStoreClient client = new BookStoreClient(args[1]);

              //Setup the call to addBook
              bookstore.Book book = null;
              book.setAuthor("Shannon Hastings");
              book.setDuey("1.1.1.1");
              book.setISBN("1234");
              book.setPDate("01-06-1999");
              book.setPublisher("Generic Publisher");
              book.setSection("1");
              book.setTitle("Programming Grid Services for Dummies");
              try{
                  client.addBook(book);
              } catch (osu.lib.bookstore.stubs.types.BookAlreadyExistsType b) {
                  System.out.println("The Book Store already has that book.");
              } catch (Exception e){
                  System.out.println("An unexpected error has occured in the call for addBook.");
              }

              //setup the call to getBook
              book = null;
              String isbn = ("1234");
              try{
                  book = client.getBook(isbn);
              } catch (osu.lib.bookstore.stubs.types.BookDoesNotExistType f) {
                  System.out.println("The book requested does not exist.");
              } catch (Exception g) {
                  System.out.println("An unexpected error has occured in the call for getBook.");
              }

            } else {
                usage();
                System.exit(1);
            }
```

Figure 3-22 BookStoreClient.java

3. Save the changes.
4. To rebuild your code, run ‘**ant all**’ from the location of the BookStore service. The location for the example is *C:\BookStore*.

Example Seven: Start the Globus Container and Run the Client

This example has progressed to a point where something can be run and tested for correctness. As a summary, the example has shown how to create a service with two implemented methods, deploy that service to Globus, and implement a client to make remote calls into the BookStore service. Before the client can be run, the Globus container must be started to expose its deployed services.

1. From the Globus location's bin directory, enter *globus-start-container.bat –nosec*. The container opens and the BookStore service is ready to receive remote calls from the client.
2. Run the client by executing *ant runClient* from the top level of the Introduce created service directory.

Example Eight: Populating caGrid Service Metadata

CaGrid users, before deploying there services to the grid, should populate the service metadata. Most of this information is automatically populated by the Introduce toolset. However, in order to fill out information that is specific to the developer and deplorer of the service, the service metadata needs to be modified by using the following steps.

1. Open the service by selecting **Introduce** > **ModifyService**. In the modification screen that opens, select the **Metadata** tab.
2. Browse to **ServiceMetadata** and select the **Edit** button. An editor window opens to enable the metadata to be edited. Populate the specific information related to the project/institution of the **ResearchCenterInfo** element.

Creating a Grid Service Using a Data Extension

This section provides the steps to set up a simple grid service using the Data Service extension in the Introduce toolkit. A Custom CQL query processor for the grid service is written, the service is deployed to a Tomcat container, and the service is invoked in a test.

The following steps are outlined:

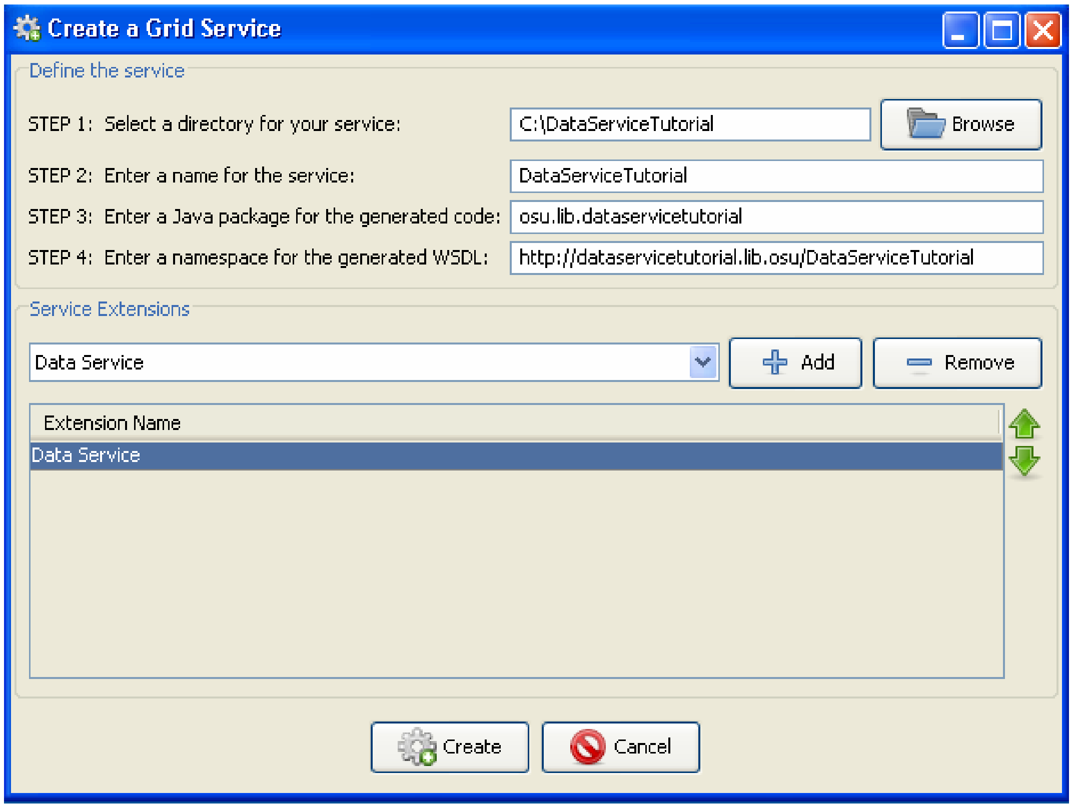
- Example One: Create a new grid service with the Data Service Extension
- Example Two: Configure the Data Service Extension
- Example Three: Implement the CQL Query Processor and the Data Service Client
- Example Four: Deploy the Data Service
- Example Five: Invoke the Data Service

Example One: Create a new Grid Service with the Data Service Extension

In this section, a new grid service is set up using the Introduce toolkit. The new service is created with the Data Service extension allowing the modification of Data Service settings within the Introduce toolkit.

1. Launch the Introduce Grid Service Authoring Toolkit.
2. Select **Create Service** from the toolbar or select **Tools->Create caBIG Service** from the menu bar. The Create caBIG Service screen displays (Figure 3-23).
3. Choose a directory in which to place the service. This tutorial uses `C:\DataServiceTutorial`.
4. Enter the Service Name. This tutorial uses `DataServiceTutorial`.
5. Choose a Package. This tutorial uses `osu.lib.dataservicetutorial`.
6. Choose a Namespace. This tutorial uses `http://dataservicetutorial.osu.lib/DataServiceTutorial`.
7. In the **Customize Service** section of the dialog, select the **Data Service** radio button.
8. Click the **Create** button at the bottom of the Creation screen.

9. In the configuration dialog that opens, select the **Custom Data Source** radio button and select **OK**. It may take a few minutes for Introduce to create the service.



STEP 1: Select a directory for your service: C:\DataServiceTutorial

STEP 2: Enter a name for the service: DataServiceTutorial

STEP 3: Enter a Java package for the generated code: osu.lib.dataservicetutorial

STEP 4: Enter a namespace for the generated WSDL: http://dataservicetutorial.lib.osu/DataServiceTutorial

Data Service

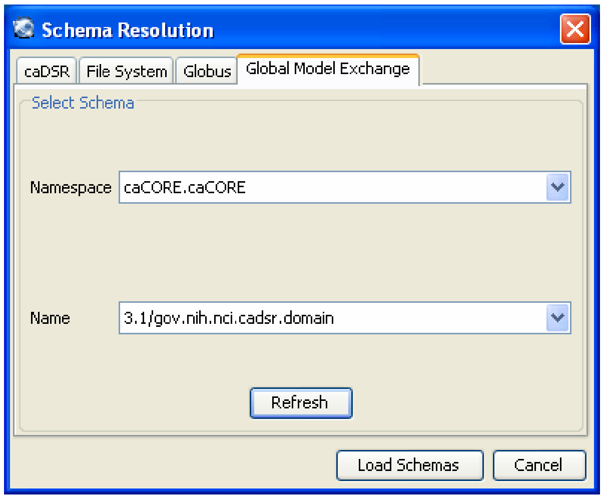
Data Service

Figure 3-23 Create caBIG Service dialog

Example Two: Configure the Data Service Extension

Next, the parameters and settings specific to the Data Service are configured. As a result of adding the Data Service Extension when creating the service, a tab is available in the Introduce service modification view where these settings are available.

1. When Introduce is finished creating the service, the Modify Service window opens. Select the **Data Service Tab** (Figure 3-24).
2. Select **caDSR Domain model** from Domain Model Source region. **Supplied Domain Model** is for users who have already generated a domain model and **No Domain Model** is for testing purposes.
3. Next, the correct package of data types is added. In the Select Data Type region, select the Project drop-down and select **caCOREs (version: 3)**. In the Package drop-down, select **gov.nih.nci.cadsr.domain**. Finally, click **Add Package**.
4. A schema resolution dialog opens informing you that the default schema for the selected package is not yet in the service. Click the **OK** button to load it. In the new dialog, select the **Global Model Exchange** tab. From the **Namespace** drop-down, select **caCORE**. From the **Name** field, select **3.1/gov.nih.nci.cadsr.domain**. Click the **Load Schemas** button to close the dialog.



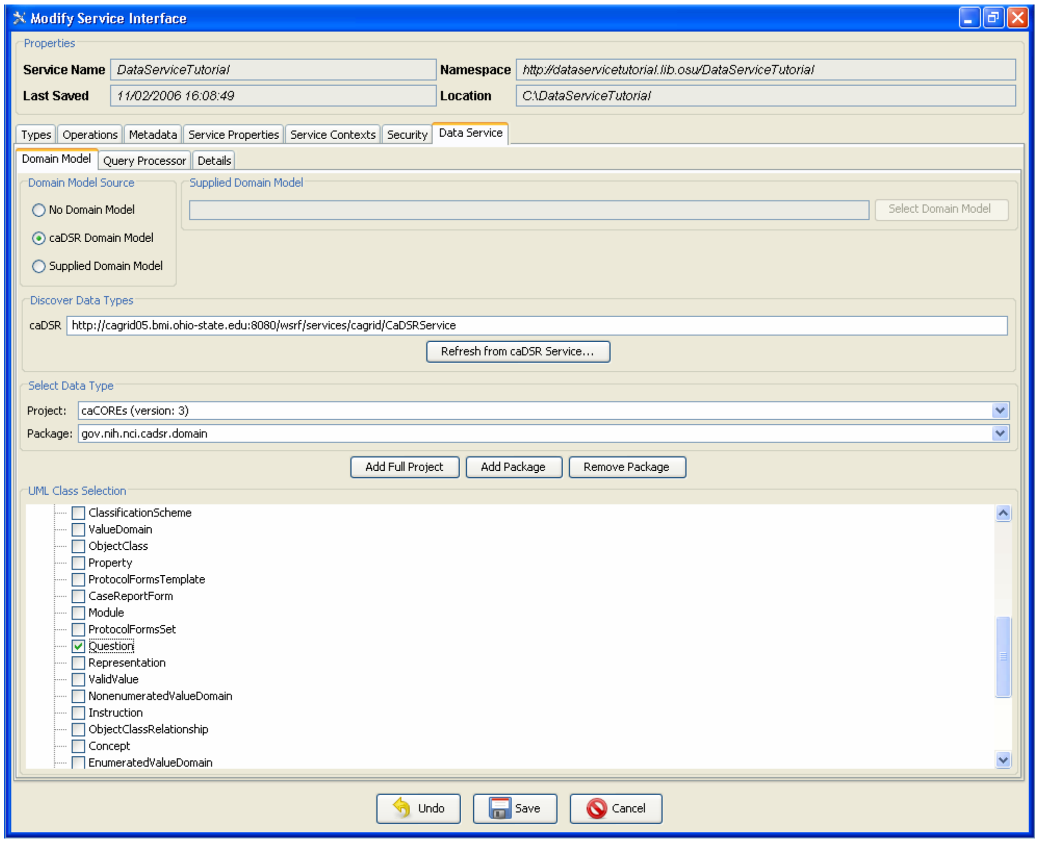
Select Schema

Namespace caCORE.caCORE

Name 3.1/gov.nih.nci.cadsr.domain

Figure 3-24 Schema Resolution dialog

5. In the UML Class Selection region, a new group should display (Figure 3-25). Expand the **gov.nih.nci.cadsr.domain** group and check the **Question** data type. Doing so specifies that the Question data type may be used in a CQL Query. In the case of this tutorial, it is the data type the client requests to be returned.



Properties

Service Name *DataServiceTutorial*
Namespace *http://dataservicetutorial.lib.osu/DataServiceTutorial*
Last Saved *11/02/2006 16:08:49*
Location *C:\DataServiceTutorial*

Domain Model Source
No Domain Model
[✓ caDSR Domain Model]
Supplied Domain Model

Supplied Domain Model

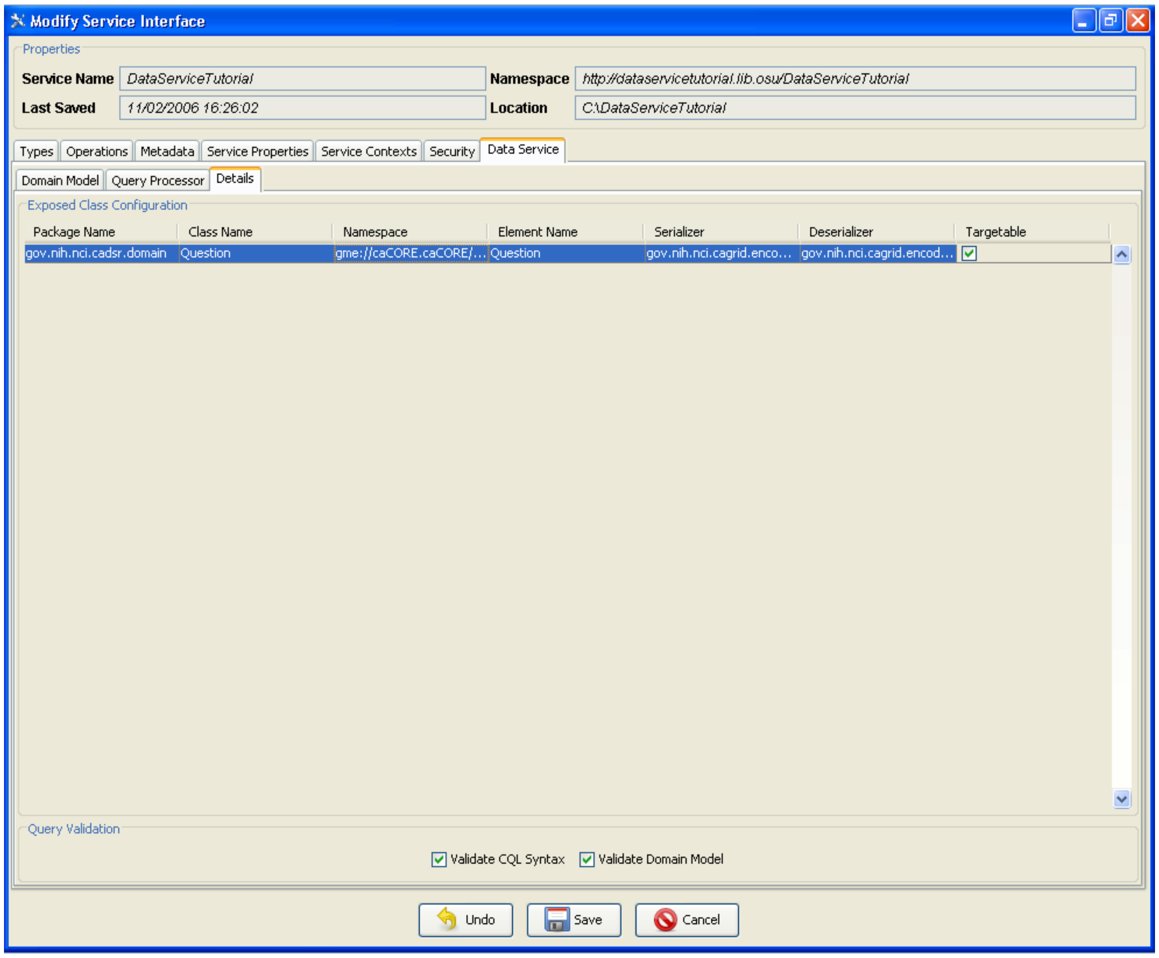
Discover Data Types
caDSR http://cagrid05.bmi.ohio-state.edu:8080/wsrf/services/cagrid/CaDSRService

Select Data Type
Project: caCOREs (version: 3)
Package: gov.nih.nci.cadsr.domain

UML Class Selection
- ClassificationScheme
- ValueDomain
- ObjectClass
- Property
- ProtocolFormsTemplate
- CaseReportForm
- Module
- ProtocolFormsSet
- [✓] Question
- Representation
- ValidValue
- NonenumeratedValueDomain
- Instruction
- ObjectClassRelationship
- Concept
- EnumeratedValueDomain

Figure 3-25 Data Service Domain Model tab

6. Now select the **Query Processor** tab. Notice the Selected Class textbox. This is the class that serves as the query processor for CQL queries made to the data service. Later in the tutorial, a method in this class is filled in to implement the processor.
7. Select the **Details** tab (Figure 3-26). Select the **Question** data type in the Exposed Class Configuration region. Right click on the **Serializer** column and select the **SDK Serialization** option. This specifies the proper serialization and deserialization of the Question object for the queries. The same serializer and deserializer should be used for any objects generated by the caCORE SDK.
8. Verify the **Targetable** option is checked. This allows queries to return this data type as the result. Data types with this option unselected are available for use in CQL queries but cannot be returned.
9. Finally, check the **Validate CQL Syntax** and **Validate Domain Model** boxes to enable error checking on the CQL queries processed at runtime.



Service Name	DataServiceTutorial	Namespace	http://dataservicetutorial.lib.osu/DataServiceTutorial
Last Saved	11/02/2006 16:26:02	Location	C:\DataServiceTutorial

Package Name	Class Name	Namespace	Element Name	Serializer	Deserializer	Targetable
gov.nih.nci.cadsr.domain	Question	gme://caCORE.caCORE/...	Question	gov.nih.nci.cagrid.enco...	gov.nih.nci.cagrid.encod...	✓

✓ Validate CQL Syntax ✓ Validate Domain Model

Figure 3-26 Data Service Details tab

10. Click **Save** at the bottom of the window. Introduce rebuilds the service with the new parameters and creates a Domain Model containing the selected data types. This process may take a few minutes depending on the internet connection.
11. Select **File->Exit** to close the Introduce toolkit.

Example Three: Implementing the CQL Processor and the Grid Service Client

This part of the tutorial implements the CQL processor and the data service client. The CQL Processor is responsible for determining what object is requested by the client, finding it in some data storage facility, and returning it to the client. Only one method is required to be implemented for the processor. This method takes a CQLQuery object containing the query information and returns a CQLResults object that contains the object or objects to be returned to the client.

1. In the folder `C:\DataServiceTutorial\src\gov\nih\nci\cagrid\dataservicetutorial\stubs\cql\`, open the file `StubCQLQueryProcessor.java`.
2. Replace the contents of the `processQuery method with the code below:`

```
CQLQueryResults results;
Mappings mappings = null;
try {
   String filename =
   ServiceConfigUtil.getClassToQnameMappingsFile();
   mappings = (Mappings) Utils.deserializeDocument(
                 filename, Mappings.class);
} catch (Exception ex) {
   throw new QueryProcessingException(
            "Error getting class to qname mappings: " +
      ex.getMessage(), ex);
}
String targetName = cqlQuery.getTarget().getName();
ArrayList resultObjects = new ArrayList();
Question question = new Question();
question.setDefaultValue("What is the meaning of life?");
resultObjects.add(question);
try
{
results = CQLResultsCreationUtil.createObjectResults(
           resultObjects, targetName, mappings);
} catch(Exception ex) {
   throw new QueryProcessingException(
                 "Error creating object results: " +
           ex.getMessage(), ex);
}
      return results;
```

3. Save StubCQLQueryProcessor.java.
4. Now open the file `DataServiceTutorialClient.java` located in the folder `C:\DataServiceTutorial\src\gov\nih\nci\cagrid\dataservicetutorial\client\`. This code directs the client's actions for the grid service. In this tutorial, the client simply sends a CQL Query requesting a Question object in return. It then prints out the defaultValue field of the object returned to it by the CQL Processor. Replace the contents of the method *main* with the code below:

```
System.out.println(
     "Running the DataServiceTutorial Service Client");
try{
if(!(args.length < 2)){
if(args[0].equals("-url")){
  DataServiceTutorial client = new
               DataServiceTutorial(args[1]);
          CQLQuery query = new CQLQuery();
          gov.nih.nci.cagrid.cqlquery.Object target =
               new gov.nih.nci.cagrid.cqlquery.Object();
          target.setName(Question.class.getName());
          query.setTarget(target);
          CQLQueryResults results = client.query(query);
           Iterator iter = new
     CQLQueryResultsIterator(results,
     DataServiceTutorial.class.getResourceAsStream(
               "client-config.wsdd"));
          while (iter.hasNext()) {
       Object o = iter.next();
           System.out.println(
               ((Question)o).getDefaultValue());
          }
  System.out.println("Query Complete.");
} else {
  usage();
  System.exit(1);
}
} else {
usage();
System.exit(1);
}
} catch (Exception e) {
e.printStackTrace();
System.exit(1);
```

5. Save DataServiceTutorialClient.java

Example Four: Deploy the Grid Service

A grid service may be deployed to either a Globus or a Tomcat container. In this tutorial, Tomcat is used.

1. Open a Command prompt and change to the `C:\DataServiceTutorial` directory.
2. Rebuild the data service code with the command: *ant all*
3. Verify that the command resulted in a Build Successful message.
4. Deploy the data service with the command: *ant deployTomcat*
   Note: To deploy to a Globus container, replace the command *deployTomcat* with *deployGlobus*.

Example Five: Test the Grid Service

Now that the service has been deployed to Tomcat, the service may be tested. An ant script located in the grid service directory runs the client and invokes the service.

1. Using a command prompt, change to the `Catalina Home` directory. Do this by entering *cd %CATALINA_HOME%*.
2. Enter */bin/startup.bat* to start the Tomcat server. Wait for a new Tomcat console window to display and finish loading Tomcat.
3. At the original command prompt, change directories to `C:\DataServiceTutorial\` and enter the command *ant runClient*.
4. The result of the run should produce the following output:

```
  C:\DataServiceTutorial>ant runClient
  Buildfile: build.xml
  setGlobus:
  checkGlobus:
      [echo] Globus: C:\progs\Globus
  defineClasspaths:
  runClient:
      [echo] Connecting to service:
http://localhost:8080/wsrf/services/cagrid/Da
  taServiceTutorial
      [java] JVM args ignored when same JVM is used.
     [java] Running the DataServiceTutorial Service Client
     [java] What is the meaning of life?
```

```
   [java] Query Complete.
BUILD SUCCESSFUL
Total time: 6 seconds
```

Creating a Data Service Using the caCORE SDK

This section provides the steps to set up a simple grid service using the Data Service extension and the caCORE data source in Introduce. The caGrid data service is set up with the help of the caCORE SDK wizard, deployed to a Tomcat container, and invoked as a test on the service. The sample service uses the caCORE SDK query processor provided with caGrid 1.0 to request Gene data types meeting certain criteria. These values, once returned, are output to the command line.

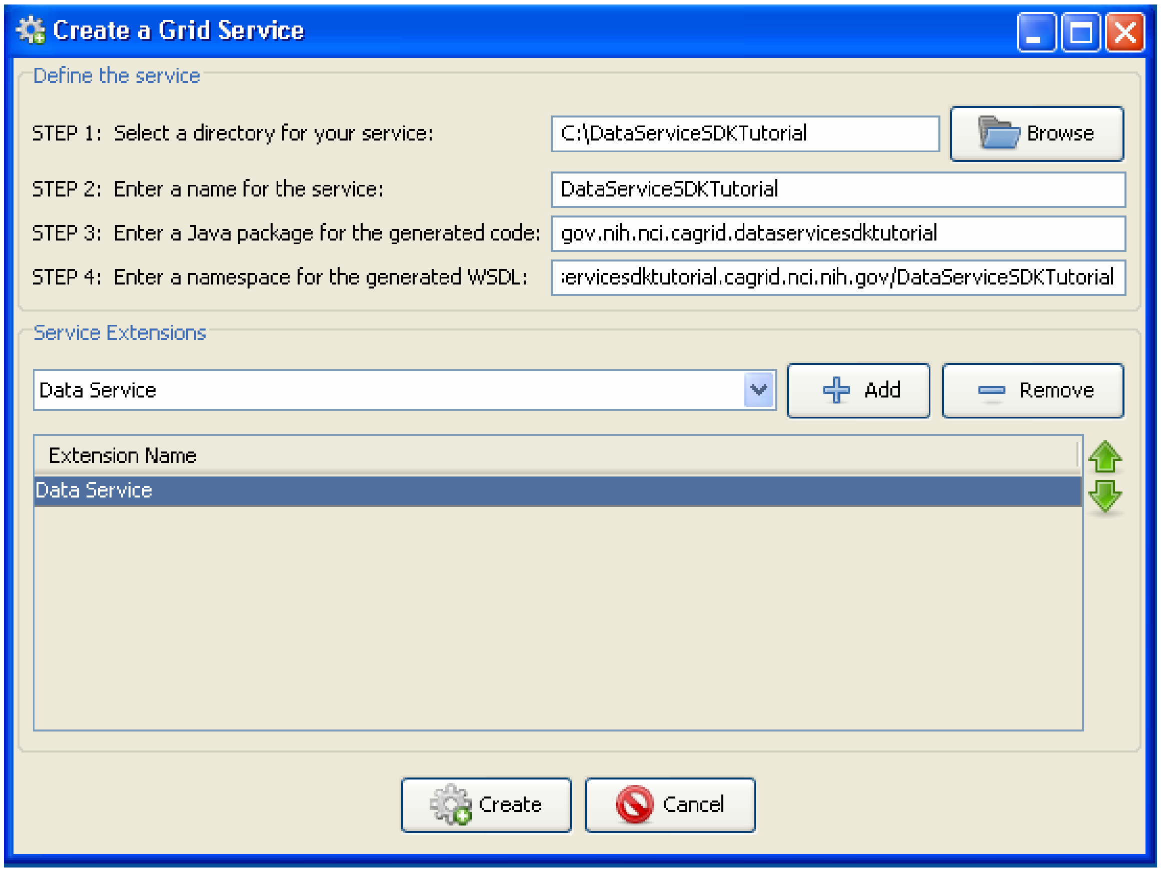
The following steps are outlined:

- Example One: Create a new grid service with the Data Service Extension
- Example Two: Use the caCORE SDK Wizard
- Example Three: Implement the Grid Service Client
- Example Four: Deploy the Grid Service
- Example Five: Test the Grid Service

Example One: Create a New Grid Service with the Data Service Extension

In this section, a new grid service is set up using the Introduce toolkit. The new service is created with the Data Service extension using the caCORE SDK Wizard.

1. Launch the Introduce Grid Service Authoring Toolkit.
2. Select **Create caBIG Service** from the toolbar or select **Tools->Create caBIG Service** from the menu. The Create caBIG Service screen opens (Figure 3-27).
3. Select a directory in which to place the generated service. This tutorial uses `C:\DataServiceSDKTutorial`.
4. Enter the Service Name. This tutorial uses `DataServiceSDKTutorial`.
5. Select a Package Name. This tutorial uses `osu.lib.datadervicesdktutorial`.
6. Select a Namespace. This tutorial uses `http://dataservicetutorial.osu.lib/DataServiceTutorial`.
7. Select the **Data Service** radio button from the Customize Service section.
8. Click the **Create** button located at the bottom of the Creation screen.
9. In the configuration dialog that opens, select the **caCORE SDK Data Source** radio button and select **OK**.
10. It may take a few minutes for Introduce to create the service.


Create a Grid Service

Define the service

STEP 1: Select a directory for your service: C:\DataServiceSDKTutorial

STEP 2: Enter a name for the service: DataServiceSDKTutorial

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

STEP 4: Enter a namespace for the generated WSDL: :ervicesdktutorial.cagrid.nci.nih.gov/DataServiceSDKTutorial

Service Extensions

Data Service

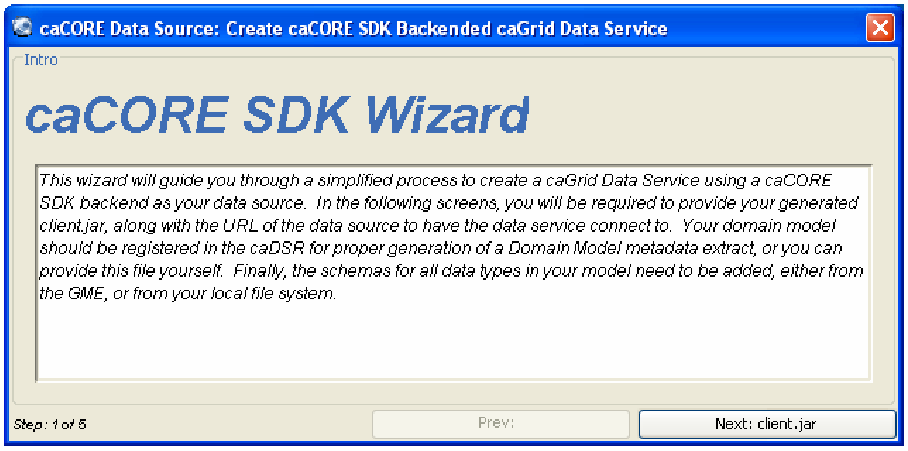
Extension Name
Data Service

Figure 3-27 Create caBIG Service dialog

Example Two: Using the caCORE SDK Wizard

The caCORE Wizard steps through five screens to enter information about the grid service. Once complete, the service is fully ready to be deployed and tested.

1. After selecting the caCORE SDK Data Source option, a dialog displays explaining the five screens of the wizard (Figure 3-28).


caCORE Data Source: Create caCORE SDK Backended caGrid Data Service

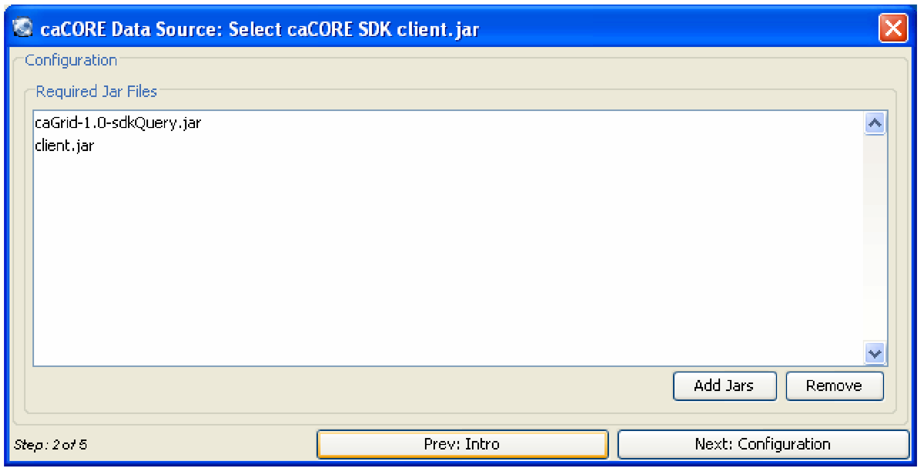
Intro

caCORE SDK Wizard

This wizard will guide you through a simplified process to create a caGrid Data Service using a caCORE SDK backend as your data source. In the following screens, you will be required to provide your generated client.jar, along with the URL of the data source to have the data service connect to. Your domain model should be registered in the caDSR for proper generation of a Domain Model metadata extract, or you can provide this file yourself. Finally, the schemas for all data types in your model need to be added, either from the GME, or from your local file system.

Figure 3-28 caCORE SDK Wizard, step 1

2. Click the ***Next: client.jar*** button to continue to the next screen of the wizard.
3. The second screen of the wizard is where jar files are added to the Data Service. Click the **Add Jars** button to open a file dialog. Find the `client.jar` file generated by the caCORE SDK and click **Open** to add it to the list (Figure 3-29).


caCORE Data Source: Select caCORE SDK client.jar

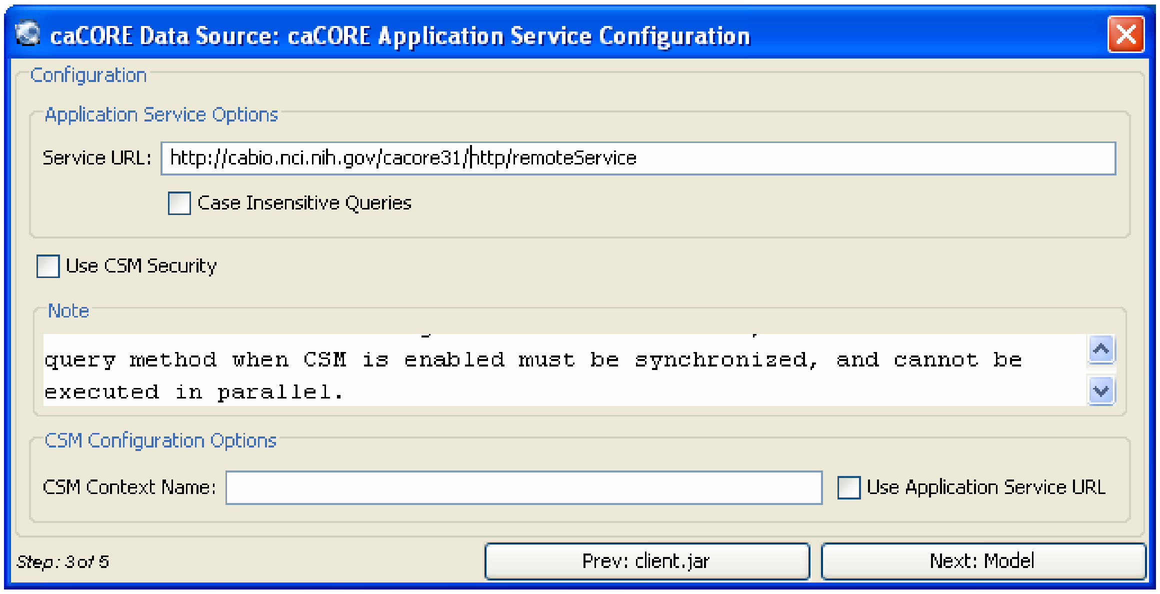
Configuration

Required Jar Files

caGrid-1.0-sdkQuery.jar
client.jar

Figure 3-29 caCORE SDK Wizard, step 2

4. The third screen of the wizard specifies the configuration options for the caCORE SDK data source (Figure 3-30). The first of these options is the service URL for the data source application. For this tutorial enter *`http://cabio.nci.nih.gov/cacore31/http/remoteService`* in the Service URL textbox. This URL is that of the NCI’s caBIO data source. The Case Insensitive Queries option is for data services that have a backend that might require queries to be performed without regard to case. Leave this option off for now. This screen also allows CSM Security to be enabled and the CSM Context Name to be changed if necessary. This tutorial does not use CSM Security.


caCORE Data Source: caCORE Application Service Configuration
Configuration
Application Service Options

Service URL: http://cabio.nci.nih.gov/cacore31/http/remoteService

Case Insensitive Queries [ ]

Use CSM Security [ ]

```
query method when CSM is enabled must be synchronized, and cannot be
executed in parallel.
```

CSM Configuration Options

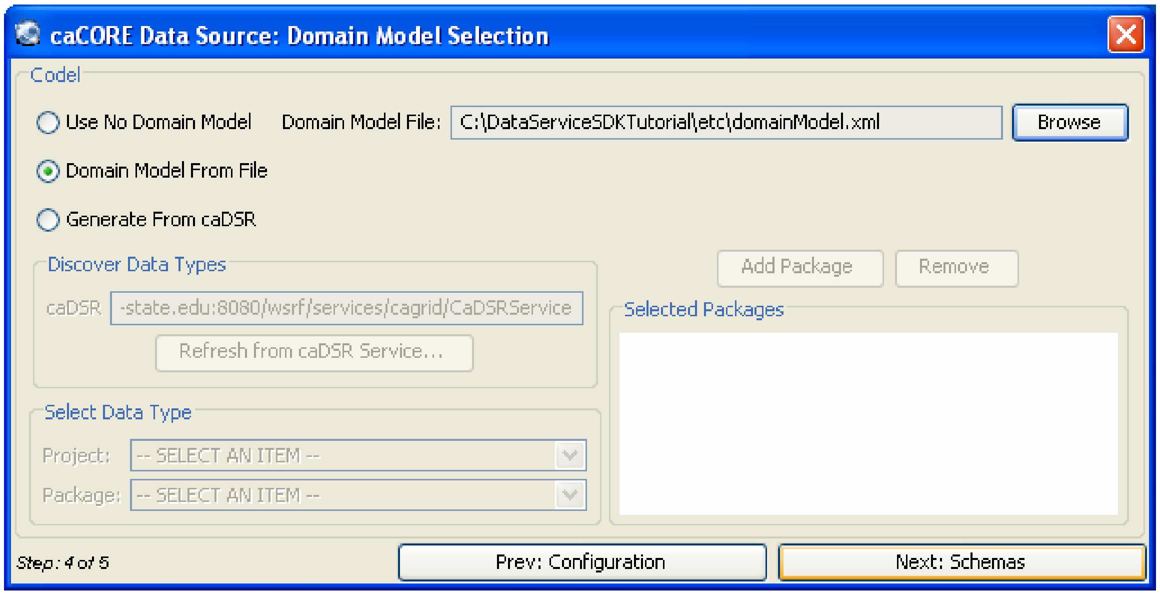
CSM Context Name:

Use Application Service URL [ ]

Step: 3 of 5
Prev: client.jar
Next: Model

Figure 3-30 caCORE Wizard, step 3

5. The fourth screen of the wizard is used to select the Domain Model (Figure 3-31). In this tutorial, the domain model is loaded from a file. Select the **Domain Model from File** radio button and click the **Browse** button on the right to navigate to the domain model file provided with this tutorial. Click **Next: Schemas** to proceed.



Use No Domain Model [✓ Domain Model From File] Generate From caDSR

Domain Model File: C:\DataServiceSDKTutorial\etc\domainModel.xml

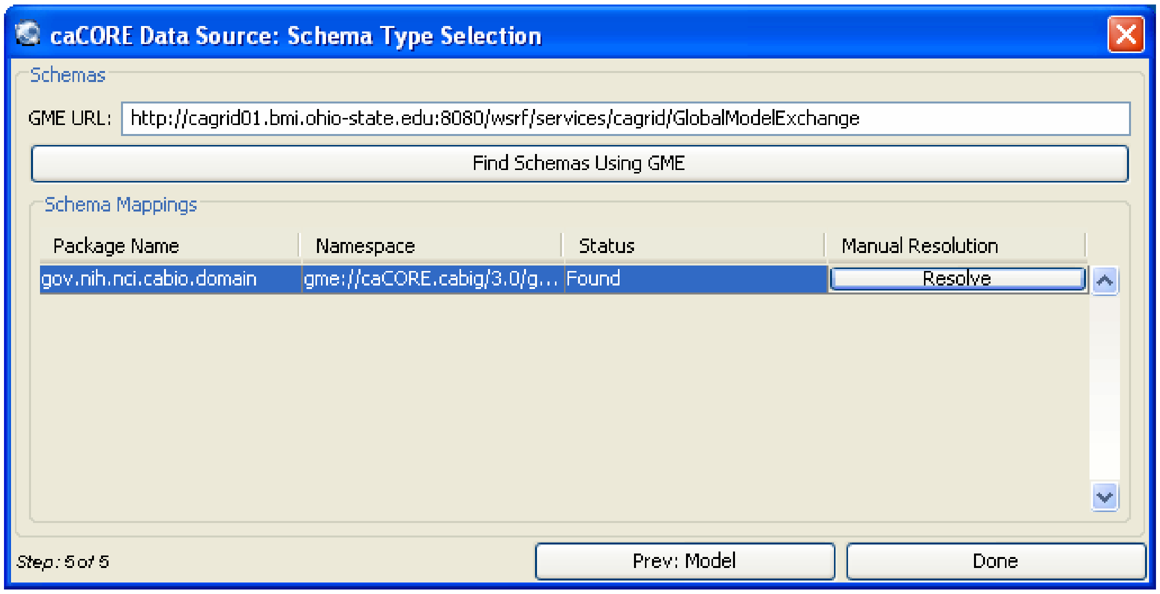
caDSR: -state.edu:8080/wsrf/services/cagrid/CaDSRService

Project: -- SELECT AN ITEM --

Package: -- SELECT AN ITEM --

Figure 3-31 caCORE Wizard, step 4

6. In the fifth screen, the caBIO schema must be resolved manually to the file provided with the tutorial (Figure 3-32). Click the **Resolve** button for the caBIO package to open the Schema Resolution Dialog (Figure 3-33). In the dialog, select the **Browse** button under the File System tab and select the caBIO schema file provided with the tutorial. Then click **Load Schemas** to close the dialog, and click **Done** to close the wizard.

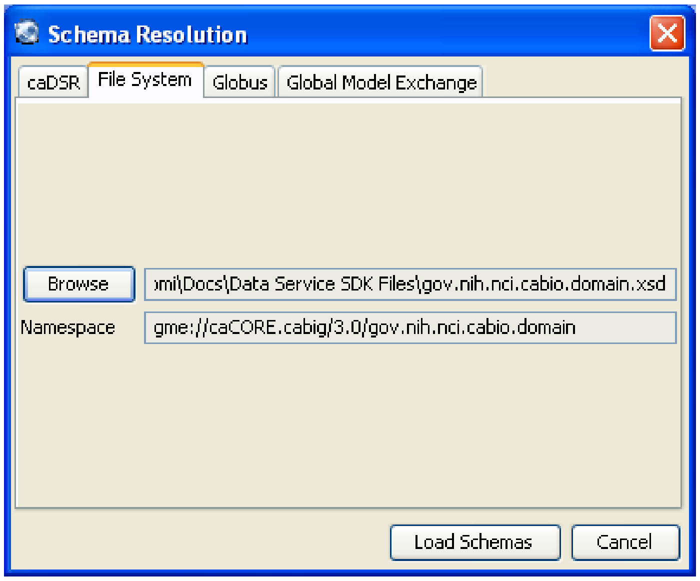


GME URL: http://cagrid01.bmi.ohio-state.edu:8080/wsrf/services/cagrid/GlobalModelExchange

Package Name	Namespace	Status	Manual Resolution
gov.nih.nci.cabio.domain	gme://caCORE.cabig/3.0/g...	Found	Resolve

Step: 5 of 5

Figure 3-32 caCORE Wizard, step 5



›mi\Docs\Data Service SDK Files\gov.nih.nci.cabio.domain.xsd

Namespace gme://caCORE.cabig/3.0/gov.nih.nci.cabio.domain

Figure 3-33 Schema Resolution dialog

Example Three: Implementing the Grid Service Client

This example implements the actions of the client to invoke the Data Service created in an earlier step. To do this, some code is added to the `DataServiceSDKTutorialClient.java` file to query the data service and output the results for verification.

1. Open the file `DataServiceSDKTutorialClient.java` located in the folder `C:\DataServiceSDKTutorial\src\gov\nih\nci\cagrid\dataservicesdktutorial\client\`. Replace the contents of the method *main* with the code below:

```
   System.out.println("Running the Grid Service Client");
   try{
         if(!(args.length < 2)){
         if(args[0].equals("-url")){
                    DataServiceSDKTutorialClient client =
                         new
                    DataServiceSDKTutorialClient(args[1]);
               gov.nih.nci.cagrid.cqlquery.CQLQuery query =
                         new
                    gov.nih.nci.cagrid.cqlquery.CQLQuery();
               gov.nih.nci.cagrid.cqlquery.Object target =
                         new
                    gov.nih.nci.cagrid.cqlquery.Object();

target.setName(gov.nih.nci.cabio.domain.Gene.class.getName());
               gov.nih.nci.cagrid.cqlquery.Attribute att =
                         new
                    gov.nih.nci.cagrid.cqlquery.Attribute();
               att.setName("symbol");
               att.setValue("BRCA%");

att.setPredicate(gov.nih.nci.cagrid.cqlquery.Predicate.LIKE);
               target.setAttribute(att);
               query.setTarget(target);
               gov.nih.nci.cagrid.cqlresultset.CQLQueryResults
results =
                         client.query(query);
               java.util.Iterator iter = new
                         gov.nih.nci.cagrid.data.utilities.CQLQue
                         ryResultsIterator(
                         results,

DataServiceSDKTutorialClient.class.getResourceAsStream(
                         "client-config.wsdd"));
               while (iter.hasNext()) {
                  Object o = iter.next();
```

```
                System.out.println(
                                ((gov.nih.nci.cabio.domain.Gene)
                        o).getFullName());
              }
              System.out.println("Query Complete.");

            } else {
              usage();
              System.exit(1);
            }
          } else {
            usage();
            System.exit(1);
          }
        } catch (Exception e) {
          e.printStackTrace();
          System.exit(1);
        }
```

2. Save the file `DataServiceSDKTutorialClient.java.`

Example Four: Deploy the Grid Service

A grid service may be deployed to either a Globus or a Tomcat container. In this example, Tomcat is used.

1. Open a Command prompt and change the directory to `C:\DataServiceSDKTutorial.`
2. Compile the grid service with the command: *ant all.*
3. Verify that the command resulted in a Build Successful message.
4. Deploy the grid service to Tomcat with the command: *ant deployTomcat.*
   Note: To deploy to a Globus container, simply replace the command *deployTomcat* with *deployGlobus*.

Example Five: Test the Grid Service

Now that the service has been deployed to the Tomcat directory, the service may be invoked. An ant script located in the grid service directory runs the client and invokes the service.

1. Using a command prompt, change to the `Catalina Home` directory by entering `cd %CATALINA_HOME%`.
2. Enter *bin/startup.bat* to start the Tomcat server. Wait for a new Tomcat console window to open and finish loading Tomcat.

3. At the original command prompt, change directories to `C:\DataServiceSDKTutorial\` and enter the command: *ant runClient*.
4. The result of the run should produce the following output:

```
C:\DataServiceSDKTutorial>ant runClient

Buildfile: build.xml

setGlobus:

checkGlobus:

   [echo] Globus: C:\progs\Globus

defineClasspaths:

  runClient:

      [echo] Connecting to service:
http://localhost:8080/wsrf/services/cagrid/Da

taServiceSDKTutorial

   [java] JVM args ignored when same JVM is used.

   [java] Running the Grid Service Client

   [java] Breast cancer 2, early onset

   [java] Breast cancer 1, early onset

   [java] Query Complete.

BUILD SUCCESSFUL
```

Chapter 4 Developing Client Applications

This chapter introduces the client applications for caGrid services.

Topics in this chapter include:

- Overview on this page
- caGrid Client APIs on page 50
- Client Application Case Study: caTRIP on page 55

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 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 4-1 details the various services and artifacts related to the description of and process for the transfer of data objects between client and service.

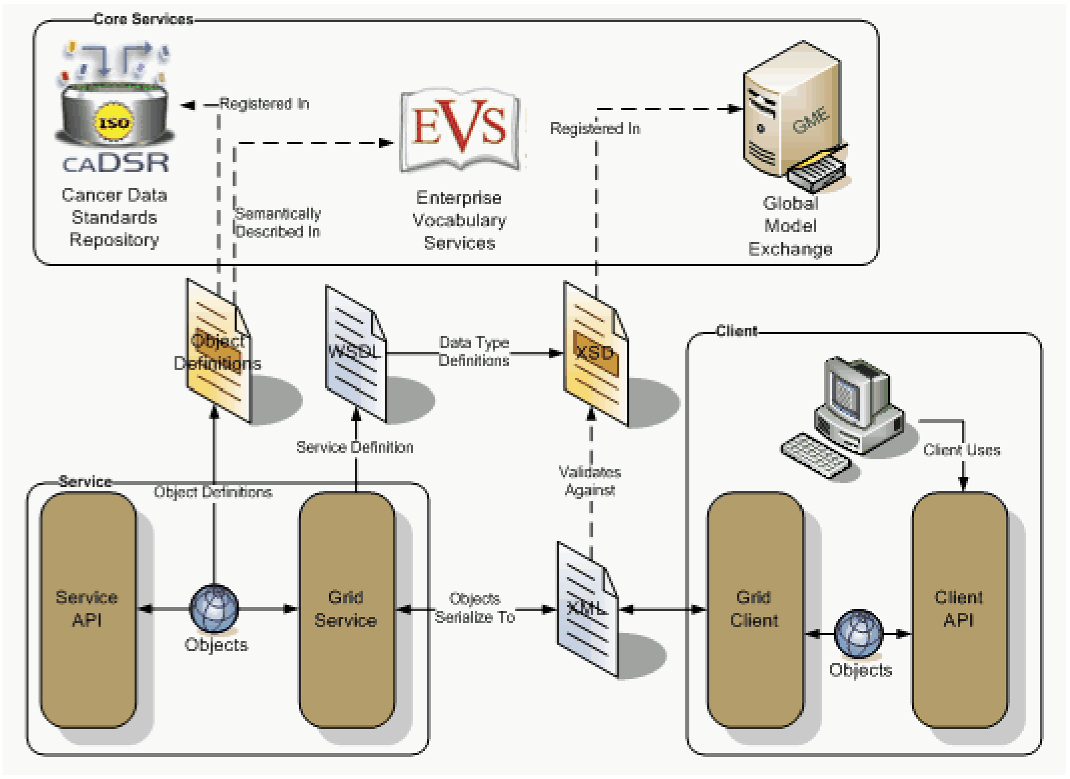


Figure 4-1 Data Description Overview

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 on 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 5 Security Administration. 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 64.

What is 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 later, 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 4-2.

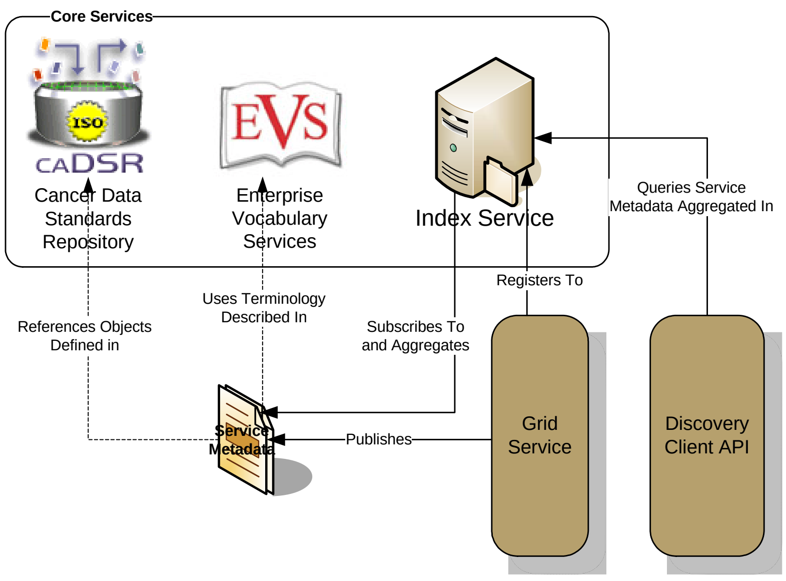

Core Services
caDSR
Cancer Data Standards Repository
EVS
Enterprise Vocabulary Services
Index Service
Queries Service Metadata Aggregated In
Registers To
Uses Terminology Described In
References Objects Defined in
Subscribes To and Aggregates
Service Metadata
Publishes
Grid Service
Discovery Client API

Figure 4-2 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 1.0 Programmer's Guide.

The simplest discovery scenario, shown in Figure 4-3, 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).

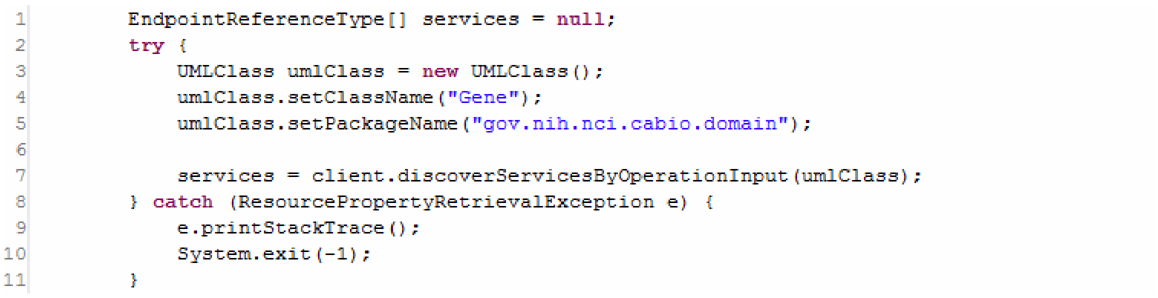
```
1        EndpointReferenceType[] allServices = null;
2        try {
3            allServices = client.getAllServices(true);
4        } catch (ResourcePropertyRetrievalException e1) {
5            e1.printStackTrace();
6            System.exit(-1);
7        }
```

Figure 4-3 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 4-4. 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 4-4 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 4-5, 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 4-5, 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 1.0 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 4-5 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 1.0 Programmer's Guide, but it is worth noting, as shown in Figure 4-2, 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 1.0 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).

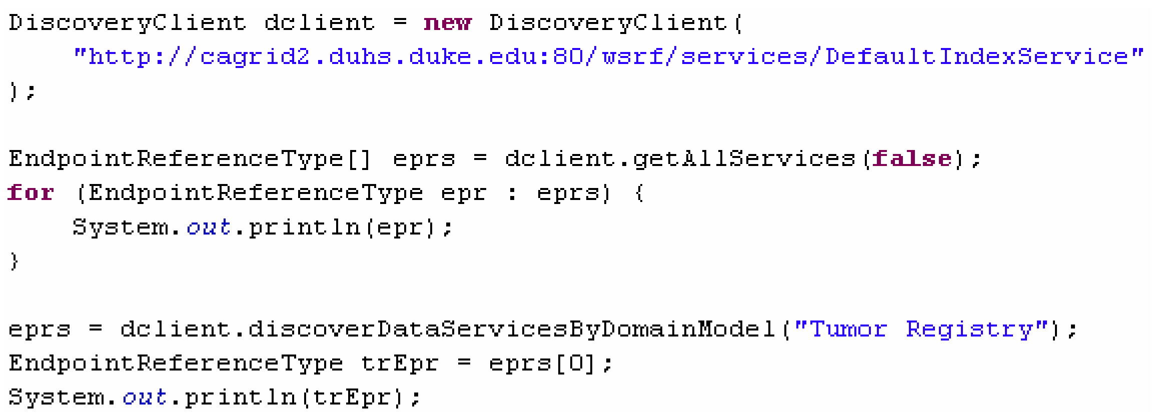
One instance where a client or application may wish to invoke operations on a service without having previously downloaded a client API is the construction of a workflow. The caGrid workflow infrastructure provides the mechanism to describe service invocations using a workflow language (BPEL), and request the workflow service perform the invocation. Further details on the workflow infrastructure can be found in the caGrid 1.0 Programmer’s Guide.

Client Application Case Study: caTRIP

The Cancer Translational Research Informatics Platform (caTRIP) project aims to solve the difficult translational research problem of outcomes analysis. This involves the querying of a number of different data elements, such as pathology biomarkers, as well as dates of diagnosis, treatment, and death. A number of different grid services can be queried in a metadata-driven manner, including caTissue CORE, CAE, the Duke Tumor Registry, and the caIntegrator SNP database. These services are exposed as caGrid 1.0 grid services. This section describes how to interact with the Tumor Registry Service from a client perspective. Three specific examples are provided: Discovery, Metadata, and Invocation.

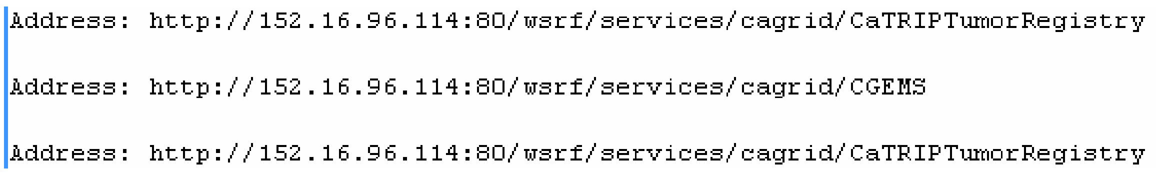
caTRIP Discovery Example

Discovery is performed by querying the Index Service for a list of Endpoint References (Figure 4-6). 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 Tumor Registry (Figure 4-7)



```
DiscoveryClient dclient = new DiscoveryClient(
    "http://cagrid2.duhs.duke.edu:80/wsrf/services/DefaultIndexService"
);

EndpointReferenceType[] eprs = dclient.getAllServices(false);
for (EndpointReferenceType epr : eprs) {
    System.out.println(epr);
}

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

Figure 4-6 caTRIP Discovery Example

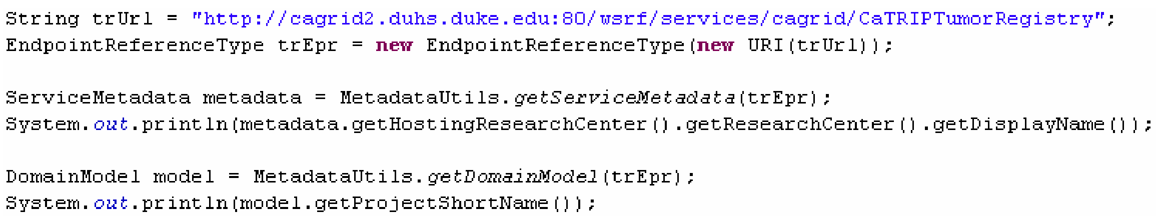


```
Address: http://152.16.96.114:80/wsrf/services/cagrid/CaTRIPTumorRegistry

Address: http://152.16.96.114:80/wsrf/services/cagrid/CGEMS

Address: http://152.16.96.114:80/wsrf/services/cagrid/CaTRIPTumorRegistry
```

Figure 4-7 Results from the caTRIP Discovery Example

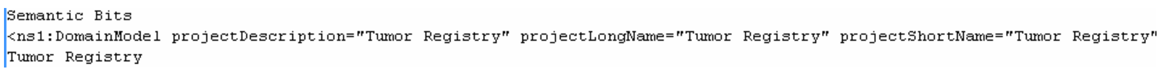
caTRIP Metadata Example

Service-level metadata can be retrieved from any caGrid 1.0 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 4-8). 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 4-9).



```
String trUrl = "http://cagrid2.duhs.duke.edu:80/wsrf/services/cagrid/CaTRIPTumorRegistry";
EndpointReferenceType trEpr = new EndpointReferenceType(new URI(trUrl));

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

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

Figure 4-8 caTRIP Metadata Example

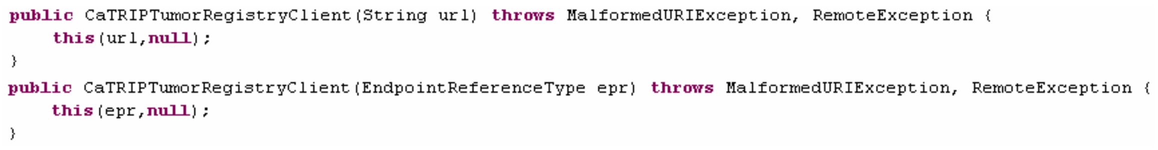


```
Semantic Bits
<ns1:DomainModel projectDescription="Tumor Registry" projectLongName="Tumor Registry" projectShortName="Tumor Registry"
Tumor Registry
```

Figure 4-9 Results from the caTRIP Metadata Example

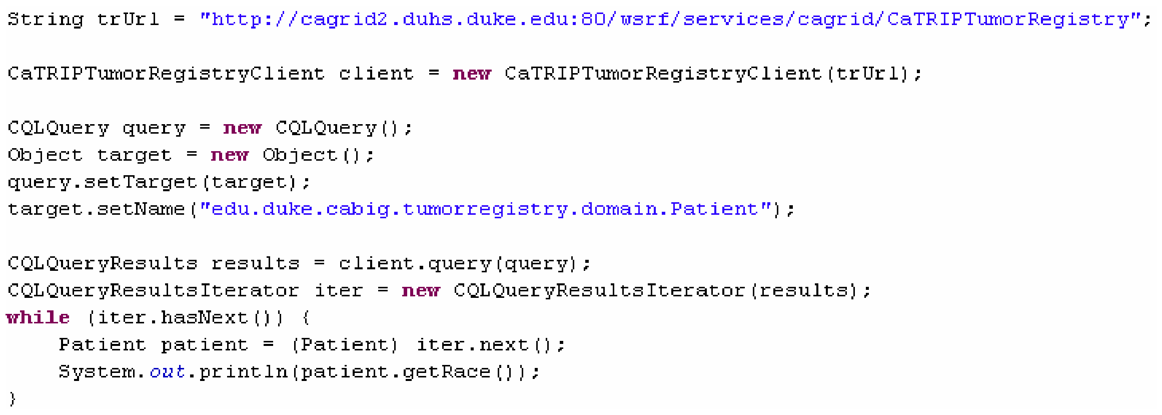
caTRIP Invocation Example

A service client can be constructed by passing in a URL to the endpoint of the service or an EPR (Figure 4-10). 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 4-11). This example demonstrates querying the Tumor Registry service for all patients and then displaying the race of each patient (Figure 4-12).



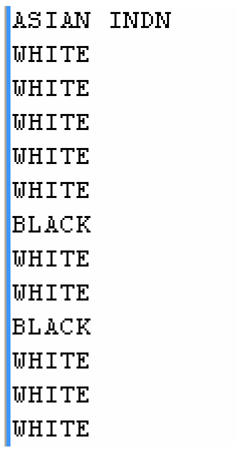
```
public CaTRIPTumorRegistryClient(String url) throws MalformedURIException, RemoteException {
    this(url,null);
}
public CaTRIPTumorRegistryClient(EndpointReferenceType epr) throws MalformedURIException, RemoteException {
    this(epr,null);
}
```

Figure 4-10 Two constructors for the caTRIP Tumor Registry data service client



```
String trUrl = "http://cagrid2.duhs.duke.edu:80/wsrf/services/cagrid/CaTRIPTumorRegistry";

CaTRIPTumorRegistryClient client = new CaTRIPTumorRegistryClient(trUrl);

CQLQuery query = new CQLQuery();
Object target = new Object();
query.setTarget(target);
target.setName("edu.duke.cabig.tumorregistry.domain.Patient");

CQLQueryResults results = client.query(query);
CQLQueryResultsIterator iter = new CQLQueryResultsIterator(results);
while (iter.hasNext()) {
    Patient patient = (Patient) iter.next();
    System.out.println(patient.getRace());
}
```

Figure 4-11 caTRIP Invocation Example



```
ASIAN INDN
WHITE
WHITE
WHITE
WHITE
WHITE
BLACK
WHITE
WHITE
BLACK
WHITE
WHITE
WHITE
```

Figure 4-12 Abbreviated results from the caTRIP Invocation Example

Chapter 5 Security Administration

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

Topics in this chapter include:

- Overview on this page
- GAARDS Administration User Interface on page 61
- Grid Trust Fabric on page 62
- Grid User Management on page 86
- Grid Grouper on page 108
- Authentication Management on page 122
- Authorization Management on page 125

Overview

The Grid Authentication and Authorization with Reliably Distributed Services (GAARDS) infrastructure serves as the caGrid 1.0 Security Infrastructure (Figure 5-1). 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.

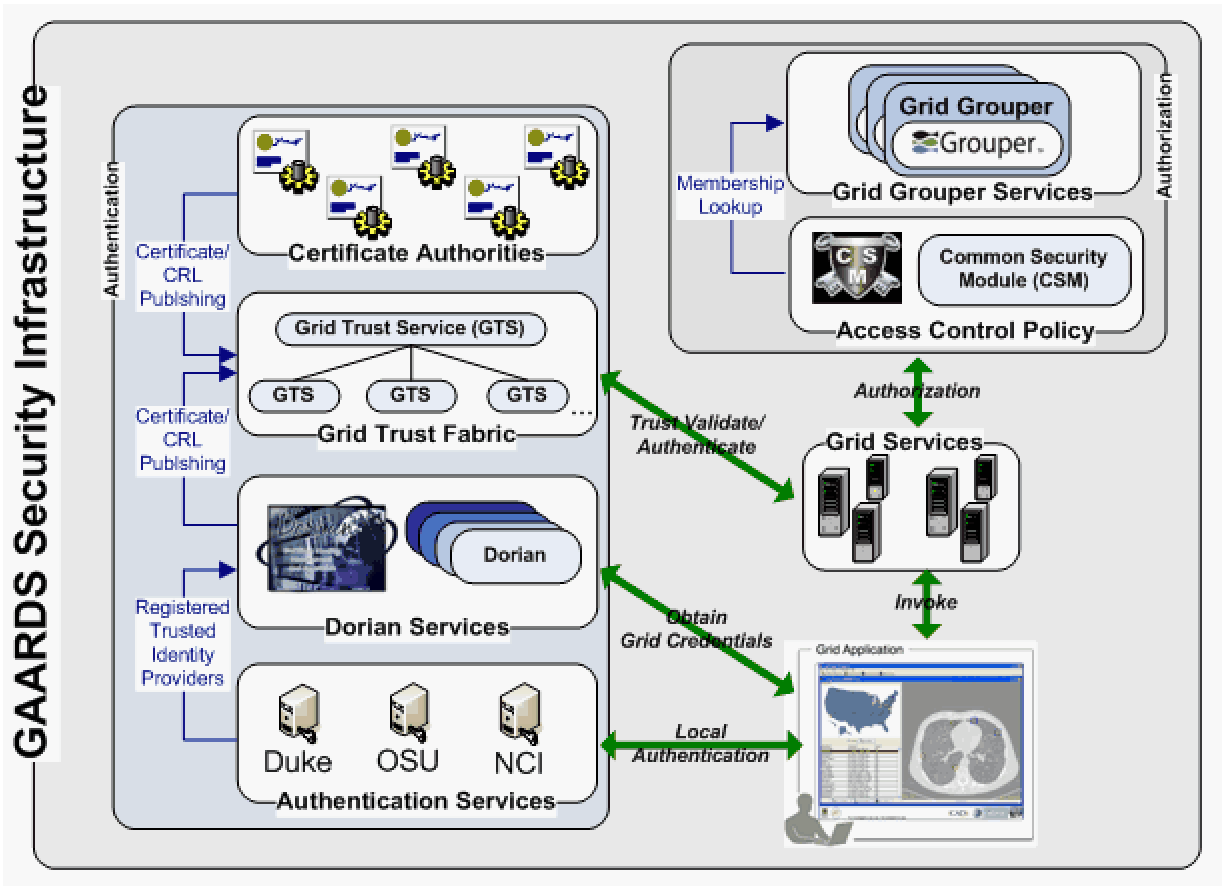


Figure 5-1 GAARDS Security Infrastructure

Figure 5-1 illustrates the GAARDS security infrastructure. In order for users/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 the 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 un-affiliated 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 5-1 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 5-1).

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 5-1, 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). The

GAARDS Admin UI can be launched 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 Trust Fabric

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 5-2 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.

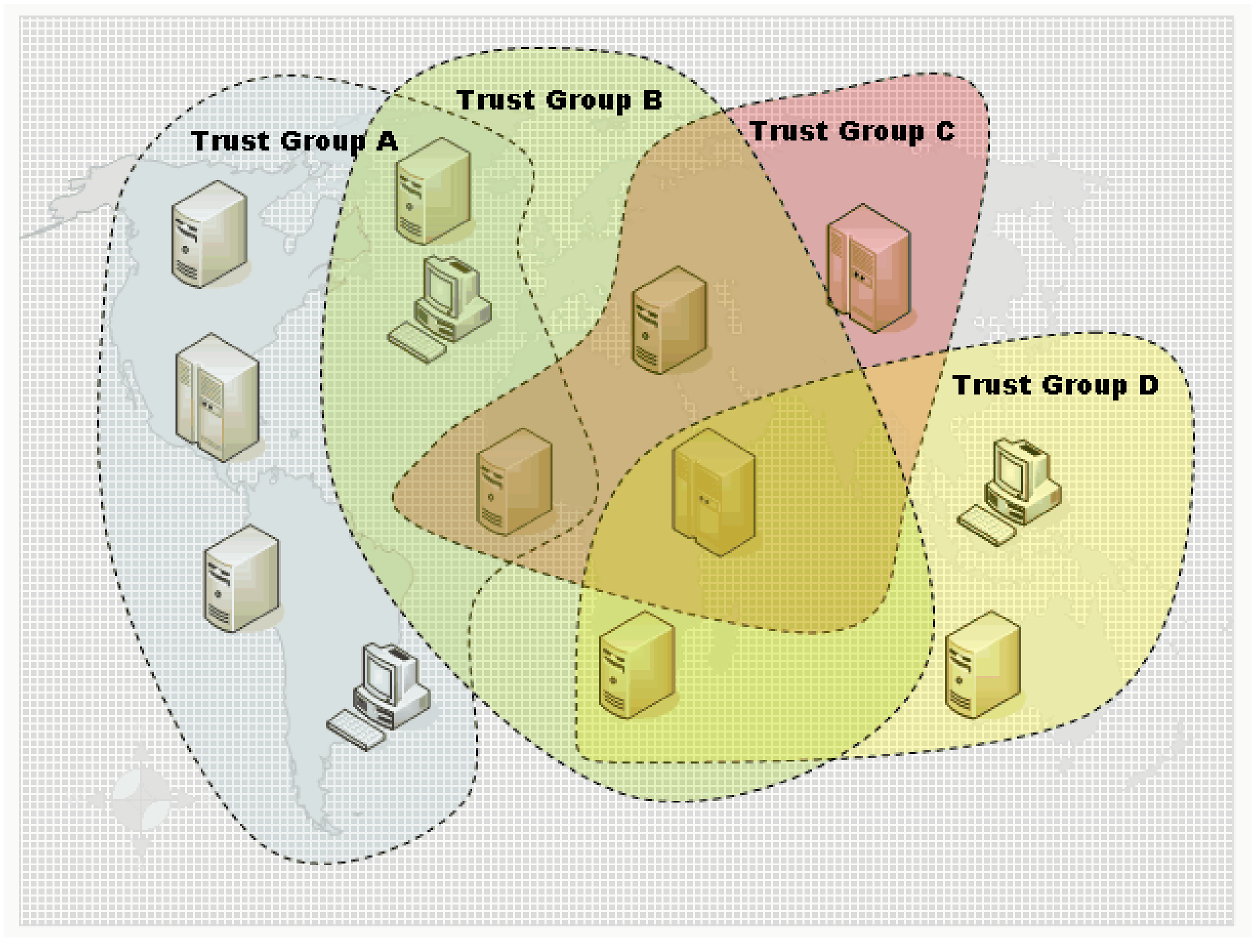


Figure 5-2 Example Grid Trust Fabric

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 X509 identity certificates. Grid Identities are assigned to users by authorities. When a grid-identity is asserted by an authority in the form of an X509 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 X509 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 is concerned with 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 X509 Identity Certificates for identifying a user. An X509 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 X509 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 5-3 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 5-3 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 is seen in the figure. In Figure 5-3, 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.

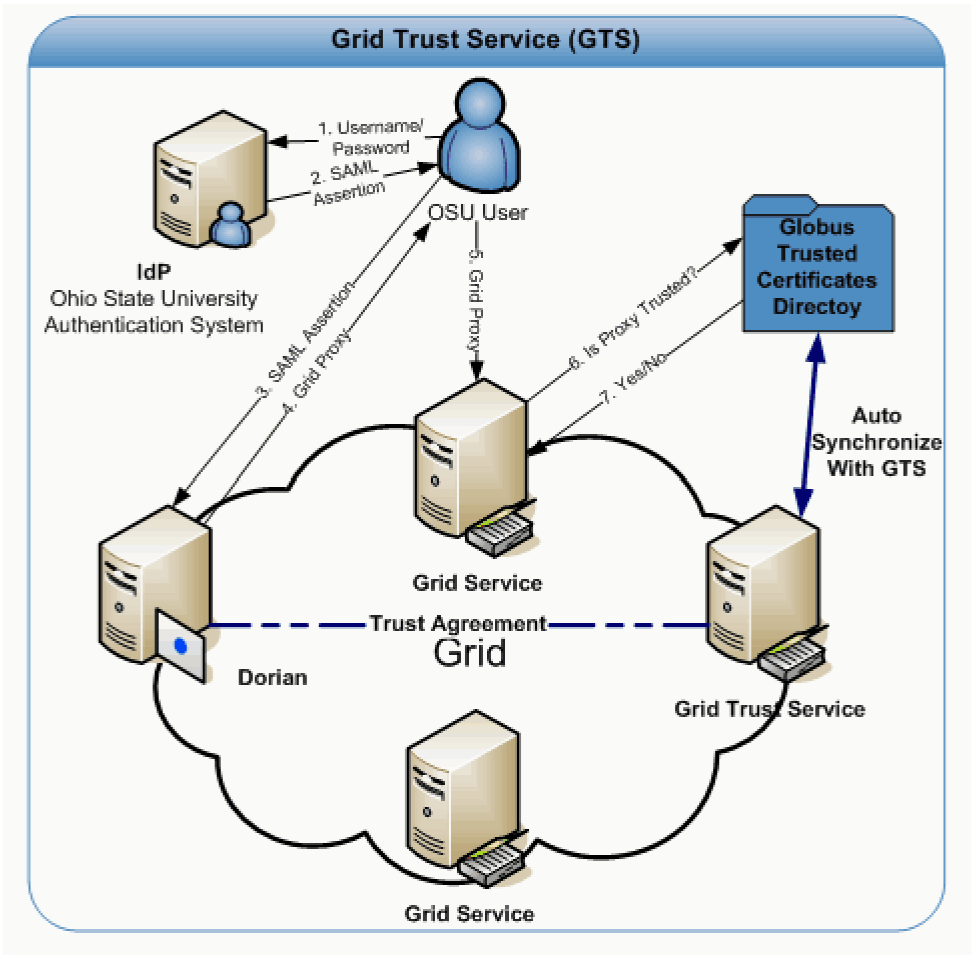


Figure 5-3 GTS Integration with Globus

Bootstrapping the Trust Fabric

As deployments leveraging the GTS to maintain the trust fabric are effectively delegating this responsibility to the GTS, it is imperative the GTS instance(s) can be trusted. Traditionally, a trust "bootstrapping" approach is adopted wherein clients and services communicating with the GTS are manually configured to trust its CA. Additionally, by default, the GTS clients perform identity authorization against the specific GTS with which they are communicating. This ensures the service providing the information about the trust fabric (the GTS) has a certificate signed by a trust authority, and that it is probably the specific instance the client intended to communicate with. There are multiple possible deployment options for assigning certificates to GTS instances. One possible way is that each GTS instance has a self-signed certificate (i.e., serving as its own CA). In such a deployment, clients and services are manually configured to trust the self-signed certificates of the GTS instances they intend to interact with. Alternatively, there can be one (or a few) *trusted root-CA(s)*, which are used to assign the certificates to each GTS instance. Installations in the grid are then bootstrapped to trust this authority or a small set of authorities. Note that even if the clients are pre-configured with the trusted CAs, the GTS infrastructure can be used as a distribution mechanism of the CA's CRLs. To examine the advantages and disadvantages of each approach, see the GTS Design Document. For the purpose of this guide, it is assume that a reasonable approach has been selected for your distribution and the CA certificates required for bootstrapping are included in your distribution. It is also assumed that SyncGTS provided with your distribution will configure Globus to trust the bootstrapping CAs before syncing.

GTS Software Prerequisites

Table 5-1 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.
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 5.0.30	The GTS can be optionally deployed as a Grid Service to a Tomcat deployed Globus Toolkit.

Table 5-1GTS Software Prerequisites

Building the GTS

To build the GTS, enter *ant clean all* from the GTS installation directory. Depending on the distribution obtained, you may be required to build from the root distribution directory to ensure that the GTS is provided with all of its dependencies.

Configuring the GTS

The GTS is configured through a single configuration file, which is located at `GTS_INSTALLATION_DIRECTORY/etc/gts-conf.xml` (Figure 5-12). 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 will 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 needs to 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>
```

Figure 5-4

The GTS can be deployed to either a HTTPS secure Globus container or an HTTPS secure Tomcat container. It is assumed that both the GTS and the container to which it is being deployed have been properly configured prior to deployment. The GTS can be deployed to a Globus container by entering *ant deployGlobus* from the GTS installation directory. Likewise the GTS can be deployed to a Tomcat container by entering *ant deployTomcat* from the GTS installation directory. It is important to note that you must add an initial administrator to the GTS before starting your container, please refer to next section *Managing Trust Fabric Administrators* section for further directions.

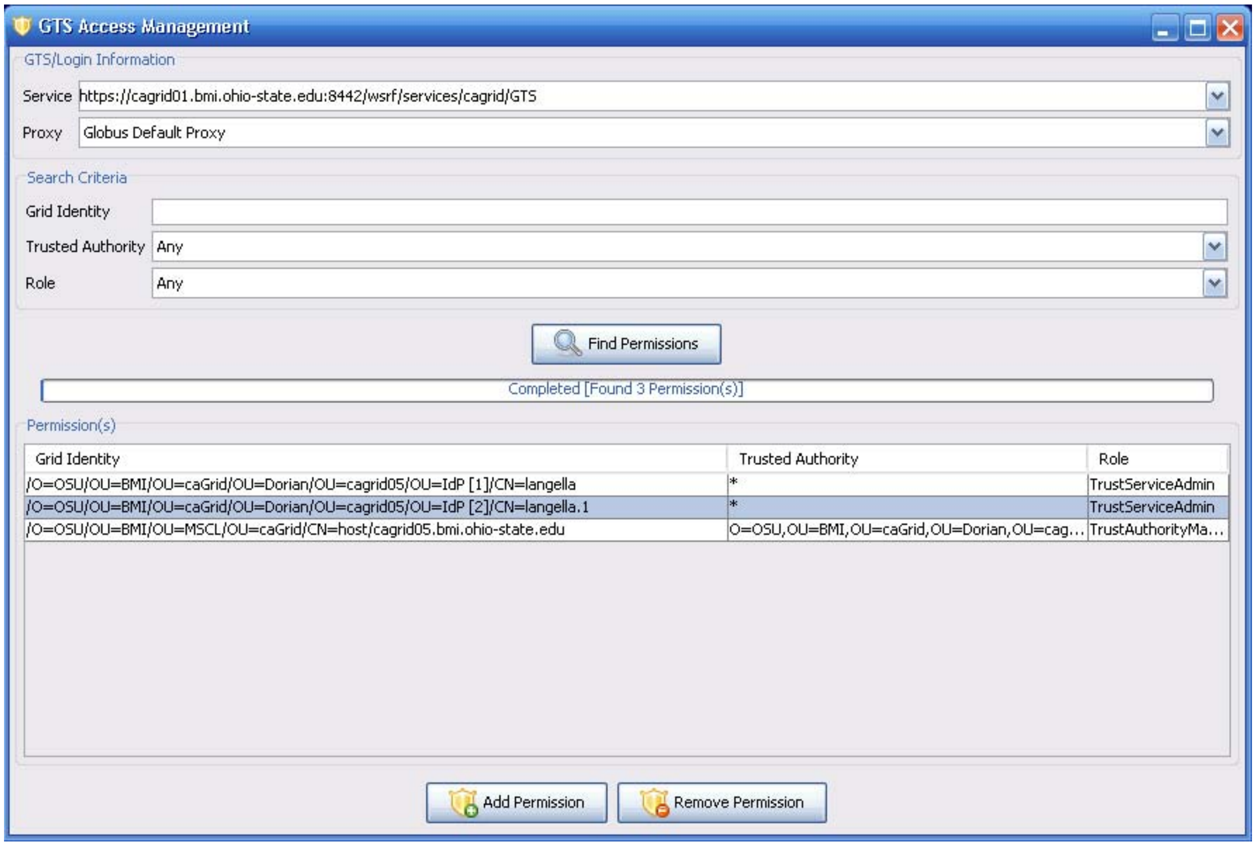
Managing Trust Fabric Administrators

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 GTS provides two methods for managing administrators: command line and the GAARDS UI. The command line approach only allows “GTS Administrators” to be added and is intended for bootstrapping the GTS with an initial administrator. The command line approach should be used to add an administrator before the GTS is started for the first time. An administrator can be added via the command line by entering *ant addAdmin* from the GTS distribution directory. The program prompts for the grid identity of the administrator to be added; entering it and pressing the ENTER key adds the requested user to the GTS as an administrator.

The GAARDS UI enables the remote management of GTS administrators. To manage administrators, the GTS requires grid credentials of a GTS administrator. To obtain grid credentials click the **Login** button on the toolbar in the GAARDS UI. For more information on obtaining grid credentials please refer to the Logging onto the Grid on page 95.

Once grid credentials are obtained, manage GTS administrators using the GAARDS UI with the following steps.

1. To start click the **Trust Management** button on the toolbar. A menu window opens. Select **Manage Access Control** and click the **Select** button.
2. The GTS Access Management window opens (Figure 5-13). To list the permissions assigned to an individual GTS, select the GTS service URI from the **Service** drop down. If the service URI you want is not in the drop down, enter it.
3. From the **Proxy** drop down, select the credentials or proxy to use to authenticate to the GTS. You may also specify search criteria to limit the permissions listed.
4. Finally to list the permissions, click the **List Permissions** button. Figure 5-13 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)]

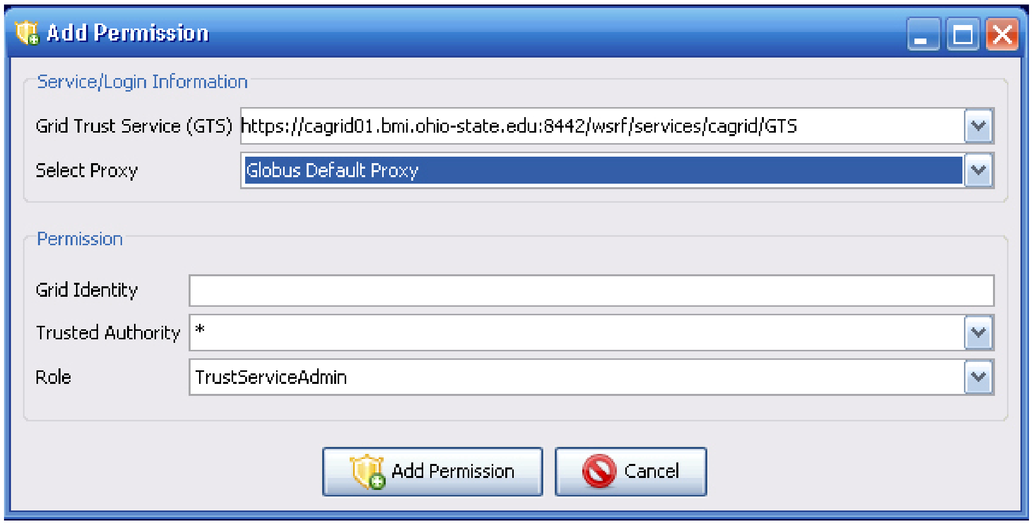
Grid Identity	Trusted Authority	Role
/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella	*	TrustServiceAdmin
/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1	*	TrustServiceAdmin
/O=OSU/OU=BMI/OU=MSCL/OU=caGrid/CN=host/cagrid05.bmi.ohio-state.edu	O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cag...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

Figure 5-13 GTS: Access Management Window

Adding a GTS Administrator

Administrators or permissions can be added by clicking the **Add Permission** button, which opens the Add Permission window shown in Figure 5-14. To add permission, use the following steps.

1. Select the service URI of the GTS in which you want to add permission. If the service URI you want to select is not in the drop down, enter it. From the **Proxy** drop down, select the credentials or proxy to use to authenticate to the GTS. In the **Grid Identity** text box, enter the grid identity of the entity being assigned the permission. From the **Trusted Authority** drop down, 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.
2. Once your selection is made, click the **Add Permission** button 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 5-14 GTS: Adding Administrators Window

Removing a GTS Administrator

To remove an administrator, select the desired permission to remove from the GTS Access Management window (Figure 5-13) and click the **Remove Permission** button.

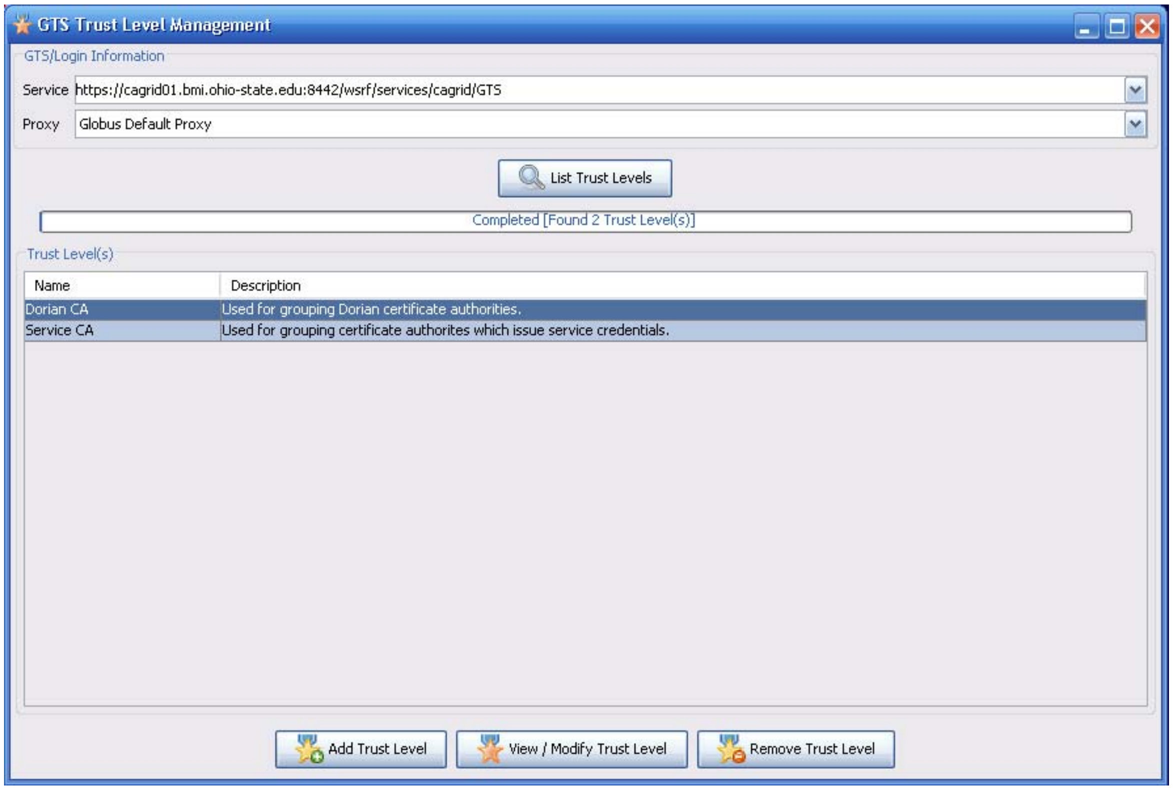
Managing Trust Levels

A 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 documentation and background check. 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 .

Trust levels can be created and managed remotely through the GAARDS UI. To manage trust levels the GTS requires grid credentials of a GTS administrator. To obtain grid credentials, click the **Login** button on the toolbar in the GAARDS UI. For more information on obtaining grid credentials, refer to Logging onto the Grid on page 95.

Once you have grid credentials, use the following steps to manage trust levels using the GAARDS UI.

1. To start, click the **Trust Management** button on the toolbar. From the menu window that opens, select **Manage Trust Levels** and click the **Select** button. The GTS Trust Level Management window opens (Figure 5-15).
2. To list the trust levels of an individual GTS, select the **GTS service URI** from the Service drop down. If the service URI you wish to select is not in the drop down, enter it. From the Proxy drop down, select the credentials or proxy to use to authenticate to the GTS.
3. Finally, click the **List Trust Levels** button and the trust levels for the selected GTS are listed as in Figure 5-15. The GTS Trust Level Management interface enables 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 5-15 GTS Trust Level Management

Viewing/Modifying Trust Levels

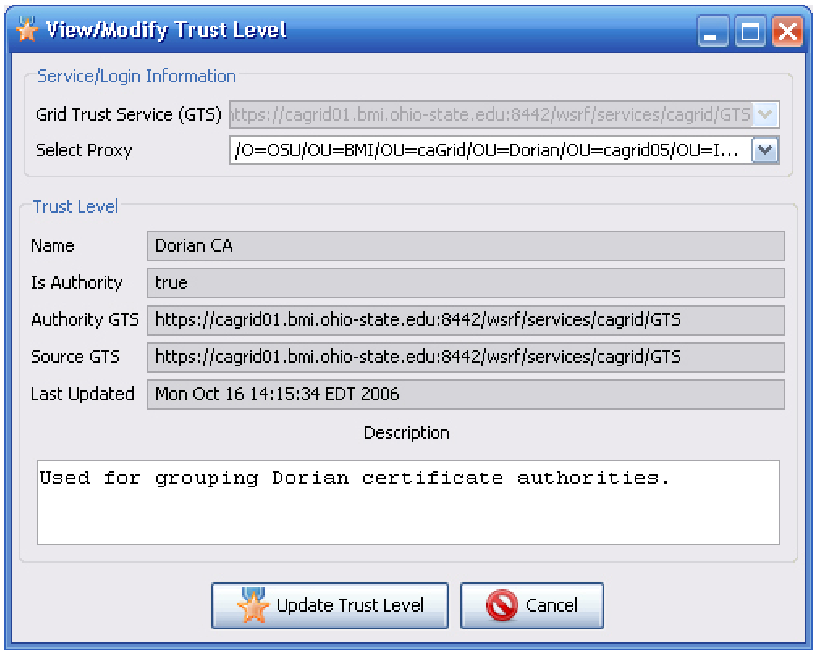
To view or modify a trust level, use the following steps.

1. Select the trust level in which you wish to view or modify and click the **View/Modify Trust Level** button. The View/Modify window opens. (Figure 5-16) for the trust level you selected.
2. 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 5-16 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.

For more details on the federated nature of GTS and on managing a federated trust fabric, refer to Managing a Federated Grid Trust Fabric on page 79.

The **Description** listing in Figure 5-16 specifies a human readable description of the trust level. 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 the **Update Trust Level** button.



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...
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

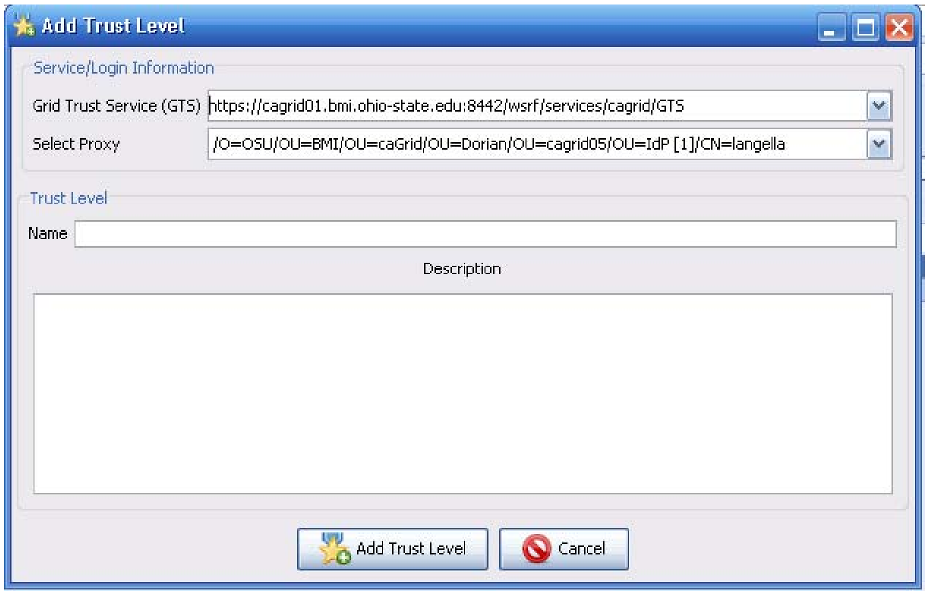
```
Used for grouping Dorian certificate authorities.
```

Figure 5-16 GTS: View/Modify Trust Level

Adding Trust Levels

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

1. Click **the Add Trust Level** button from the GTS Trust Level Management Window (Figure 5-15). The **Add Trust Level** window opens (Figure 5-17).
2. To add a trust level, first select the URL of the GTS to which the trust level will be added. Then specify the name and description of the trust level.
3. Finally click the **Add Trust Level** button 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

Name

Figure 5-17 GTS: Add Trust Level

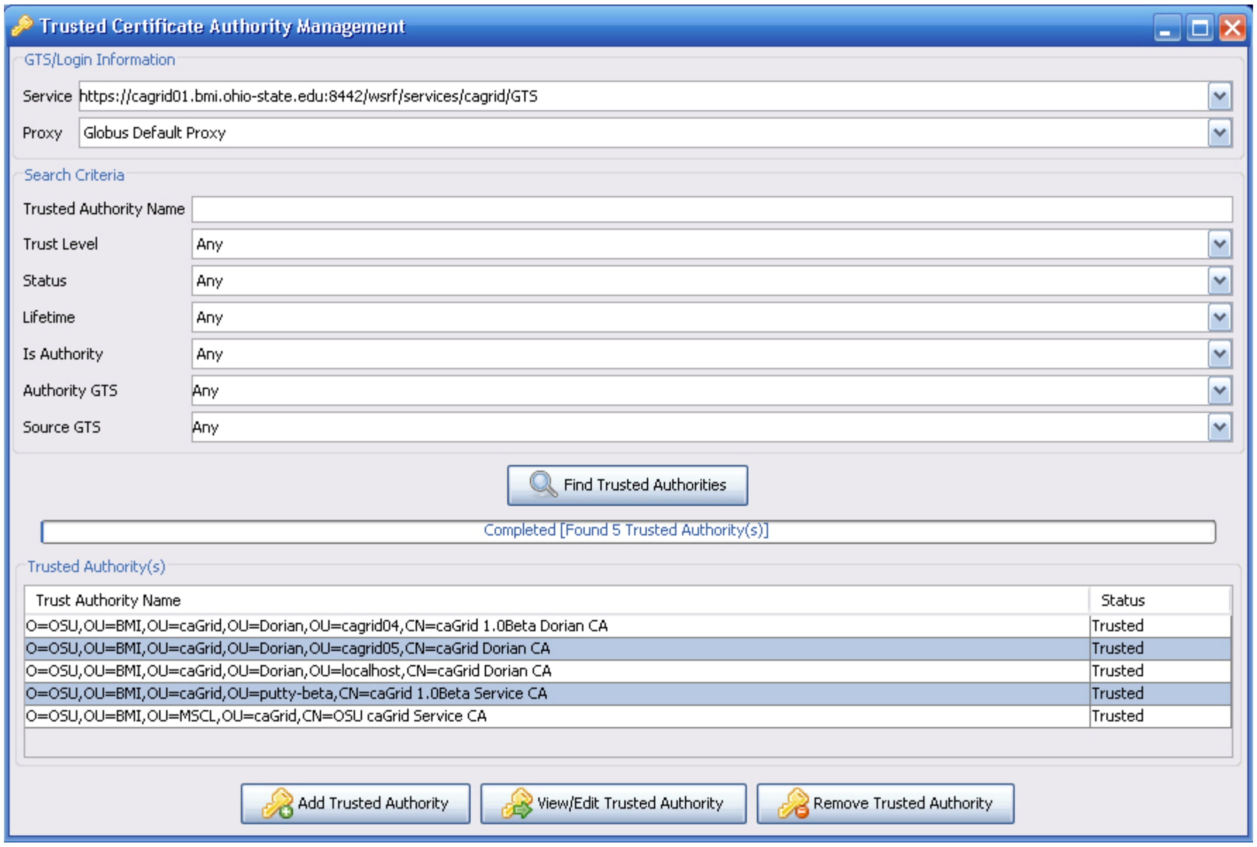
Removing Trust Levels

To remove a trust level, select the trust level to remove from the **GTS Trust Level Management** window (Figure 5-15) and click the **Remove Trust Level** button.

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 (Figure 5-18). 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 grid credentials of a GTS administrator.

To obtain grid credentials, click the **Login** button on the toolbar in the GAARDS UI. For more information on obtaining grid credentials, refer to Logging onto the Grid on page 95.



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 5-18 GTS: Trusted Certificate Authority Management

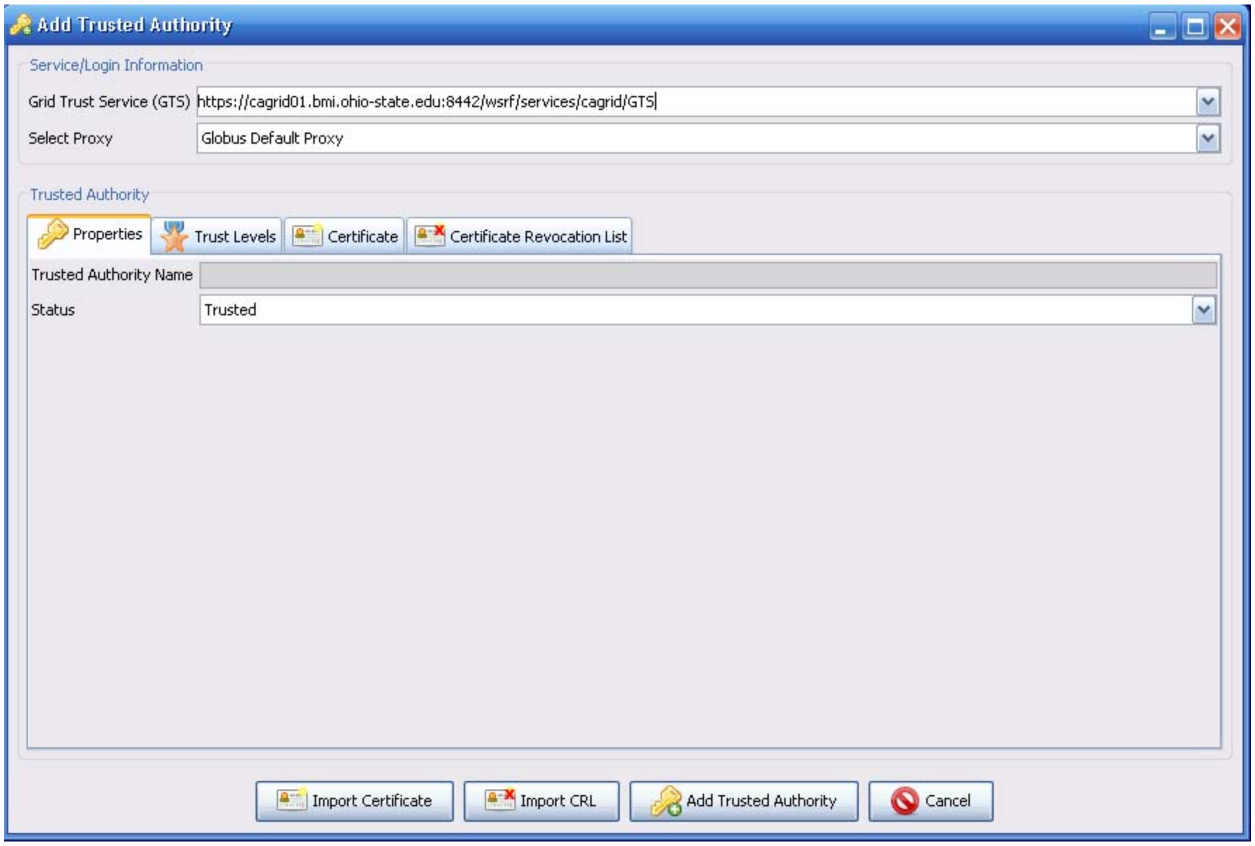
To manage certificate authorities, use the following steps.

1. Launch the **Trusted Certificate Authority Management** window and click the **Trust Management** button on the toolbar.
2. Select the **Manage Trusted Authorities** option from the menu window that opens and click the **Select** button.
3. The **Trusted Certificate Authority Management** window provides a method of discovery/searching for trusted certificate authorities in a GTS. The GTS supports a wide array of search criteria for discovery trusted certificate authorities. To discover trusted certificate authorities, simply select the GTS you wish to search and specify your search criteria.
4. Finally, click **Find Trusted Authorities** button. 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 CA certificate.

Adding Trusted Certificate Authorities

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. Click the **Add Trusted Authority** button from the **Trusted Certificate Authority Management** window (Figure 5-18). The **Add Trusted Authority** window opens (Figure 5-19).
2. To add a CA, you must provide the CA certificate in PEM format. Click the **Import Certificate** button to prompt you to specify the location of the CA certificate. Once you have specified the CA certificate, the Trusted Authority Name is set to the subject DN of the certificate you specified. If you click **Certificate** tab, the details of the imported certificate display.
3. Next specify the level of trust that should be placed in this certificate authority. Click the **Trust Levels** tab. The trust levels supported by the GTS display. Select the trust levels that you wish to assign to this CA. By default the status of the CA is set to "Trusted".
4. To change this, click the **Properties** tab. Optionally, if the CA has a CRL, you may import it. Like the CA certificate the CRL must be in PEM format. To import the CA's CRL, click the **Import CRL** button. This prompts you to specify the location of the CRL. Click the **Certificate Revocation List** tab to show you the details of the CRL.
5. Once the required information is provided, submit the CA to the GTS by clicking the **Add Trusted Authority** button.



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 5-19 GTS: Add Trusted Authority

Viewing/Modifying Trusted Certificate Authorities

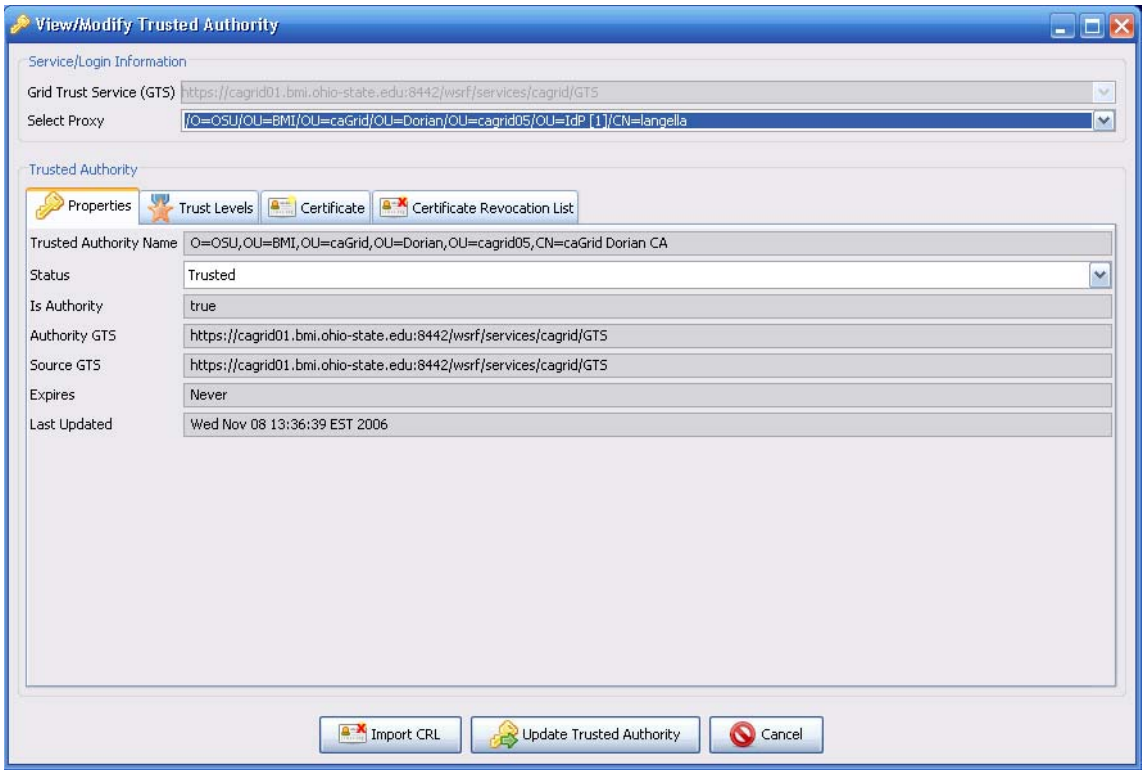
To view or modify a Trusted Certificate Authority, use the following steps.

1. From the **Trusted Certificate Authority Management** window (Figure 5-18), select the certificate authority to view/modify and click the **View/Modify Trusted Authority** button.
2. The **View/Modify Trusted Authority** window opens (Figure 5-20), which is a window containing four tabs.
   - 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. For more details on the federated nature of GTS and on managing a federated trust fabric, refer to

Managing a Federated Grid Trust Fabric on page 79. The *"Is Authority"* field specifies whether or not the GTS is the authority for the certificate authority. The *"Authority GTS"* field specifies the URL of the authority GTS for the certificate authority. The *"Source GTS"* listing specifies the URL of the GTS in which the GTS discovered or obtained the certificate authority from. 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 that last time the certificate authority listing was updated.

- The **Trust Levels** tab contains a listing of all the trust level 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 an un-selecting 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. This prompts you to specify the location of the CRL on the file system. It is important to note that the CRL must be in PEM format.

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



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 5-20 GTS: View/Modify Trusted Authority

Removing Trusted Certificate Authorities

To remove a Trusted Certificate Authority, select the certificate authority to remove from the **Trusted Certificate Authority Management** window (Figure 5-18) and click the **Remove Trusted Authority** button.

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 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. GTSs 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 5-21 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.

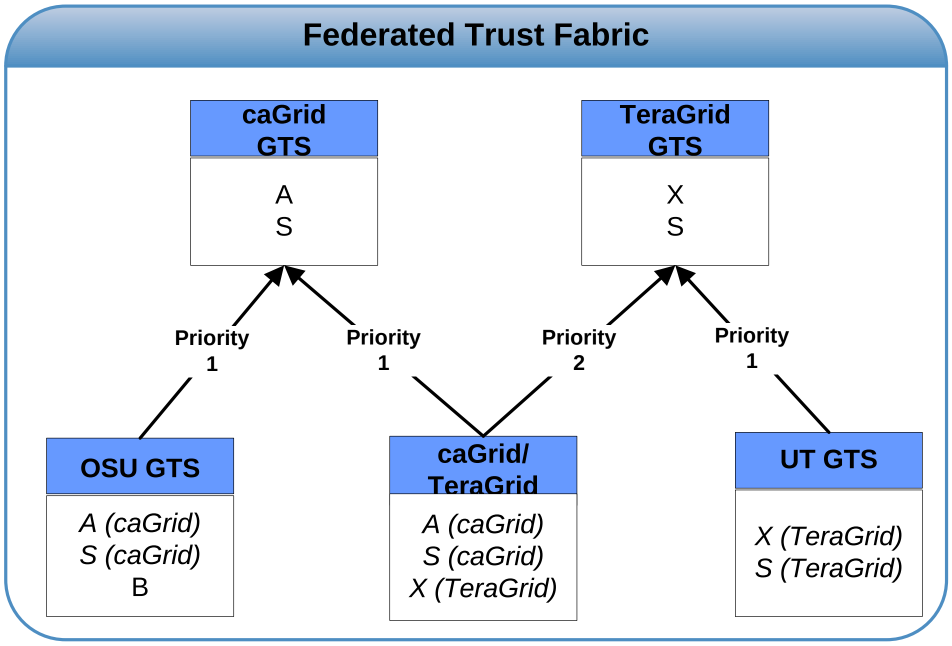


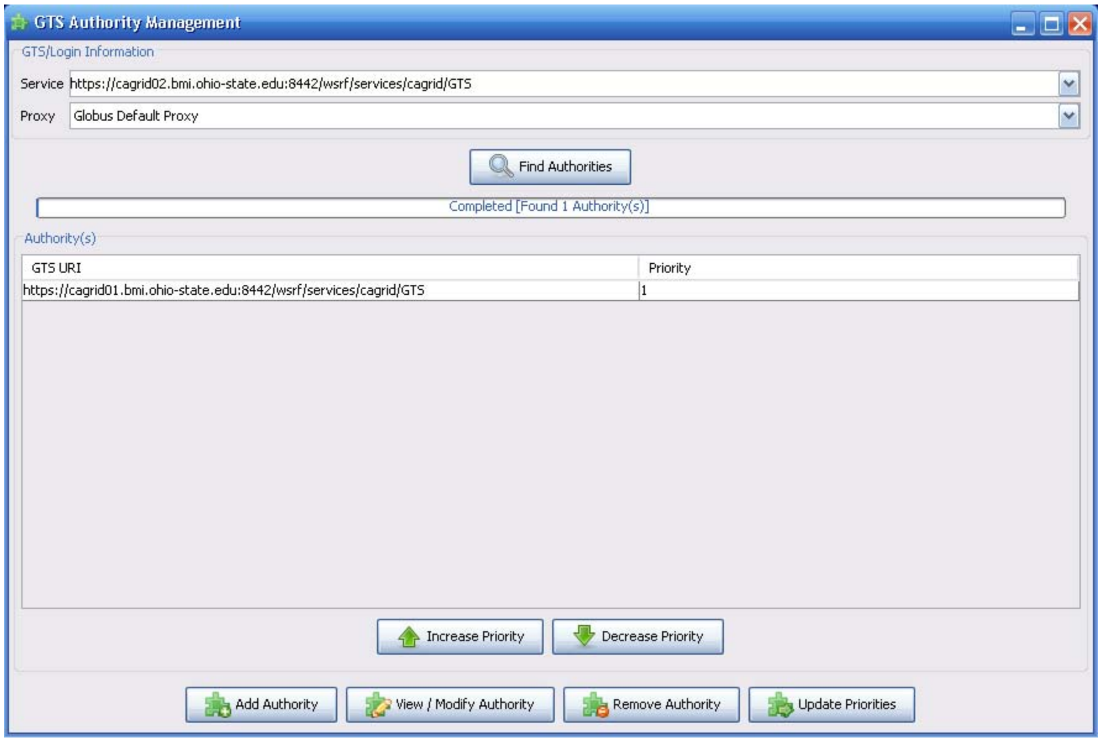
Figure 5-21 Federated Grid Trust Fabric

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. Authority GTS(s) are managed through the **GTS Authority Management** window (Figure 5-22). You must be a GTS administrator to manage Authority GTSs, thus grid credentials are required.

To obtain grid credentials, use the following steps.

1. Click the **Login** button on the toolbar in the GAARDS UI. For more information on obtaining grid credentials, refer to Logging onto the Grid on page 95.
2. To launch the **GTS Authority Management w**indow, click the **Trust Management** button on the toolbar. Select **Manage Authorities** from the menu window that opens and click the **Select** button.
3. To list all the authorities for a GTS, select the **GTS URI** from the **Service** drop down. Next select the grid credentials that should be used to authenticate to the GTS from the **Proxy** drop down.
4. Finally, click the **Find Authorities** button to list all the Authority GTSs in the table below the progress bar. Each Authority GTS is listed with its service URL and 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)]

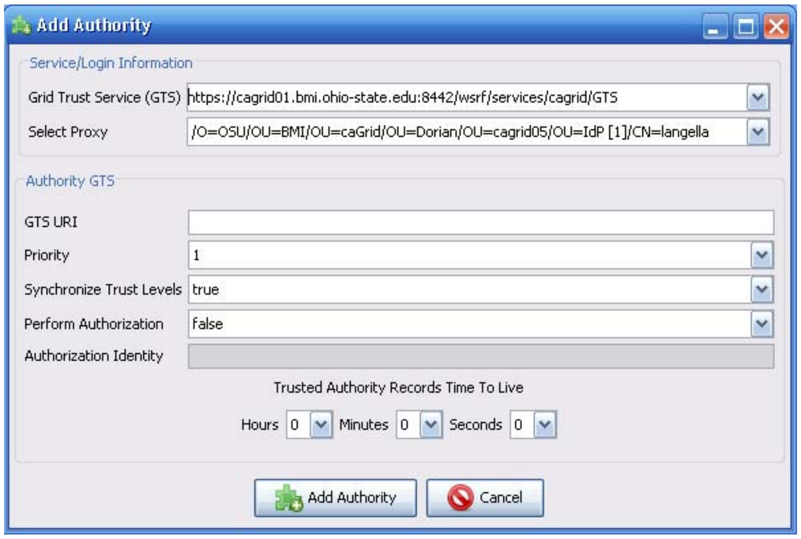
GTS URI	Priority
https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS	1

Figure 5-22 GTS: Authority Management

Adding an Authority GTS

To add an Authority GTS, use the following steps.

1. Click the **Add Authority** button, which opens the **Add Authority** window. Adding an Authority GTS requires specifying the URI of the authority GTS.
2. Next specify a priority with respect to other Authority GTSs. Then specify whether or not the GTS should synchronize or inherit the trust levels from its Authority GTS. Also 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 provided. 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.



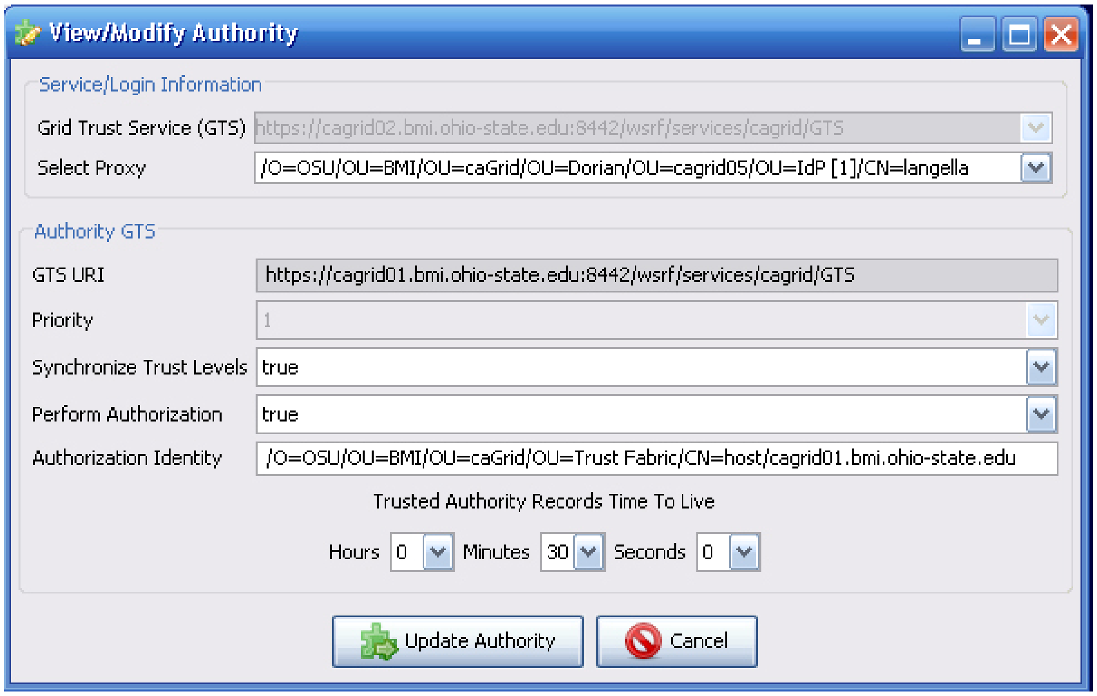
Grid Trust Service (GTS) https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Select Proxy /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella
GTS URI
Priority 1
Synchronize Trust Levels true
Perform Authorization false
Authorization Identity
Trusted Authority Records Time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

Figure 5-23 GTS Add Authority

Updating an Authority GTS

To update an Authority GTS, use the following steps.

1. Click the **Update Authority** button on the **GTS Authority Management** window (Figure 5-22), which opens the **Update Authority** window.
2. 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 However, the priorities of Authority GTSs may be updated through the **GTS Authority Management** window (Figure 5-22).


View/Modify Authority
Service/Login Information
Authority GTS
Trusted Authority Records Time To Live
Update Authority
Cancel

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
Hours 0 Minutes 30 Seconds 0

Figure 5-24 GTS: Update Authority

Prioritizing an Authority GTS

To prioritize the Authority GTSs, use the following steps.

1. In the **GTS Authority Management** window (Figure 5-22), select the desired Authority GTS to change the priority of. Select the **Increase Priority** button to increase the priority of the selected Authority GTS or select the **Decrease Priority** button to decrease the priority of the selected Authority GTS. Remember the Authority GTS with the lowest number has the highest priority.
2. Once the priorities of the Authorities GTSs are organized properly, select the **Update Priorities** button to commit the priorities to the GTS.

Removing an Authority GTS

To remove Authority GTSs through the **GTS Authority Management** window (Figure 5-22), select the desired Authority GTS to remove and click the **Remove Authority** button.

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 synched with the trust fabric.

The SyncGTS tool provides three methods for syncing with the trust fabric: 1) Command Line, 2) Service Based, and 3) Programmatic. For the purpose of this guide, directions are provided for the command line and service based approaches.

Synchronization Description

SyncGTS uses an XML file to describe what and how to synchronize the local environment with the trust fabric. This XML file is referred to as `sync description` and is located at:

SyncGTS_Installation _Directory/etc/sync-description.xml

Figure 5-24 provides a graphical overview of the XML schema representing the sync description. The root element *SyncDescription* contains four child elements: *SyncDescriptor*, *Excluded CAs, DeleteInvalidFiles,* and *NextSync.*

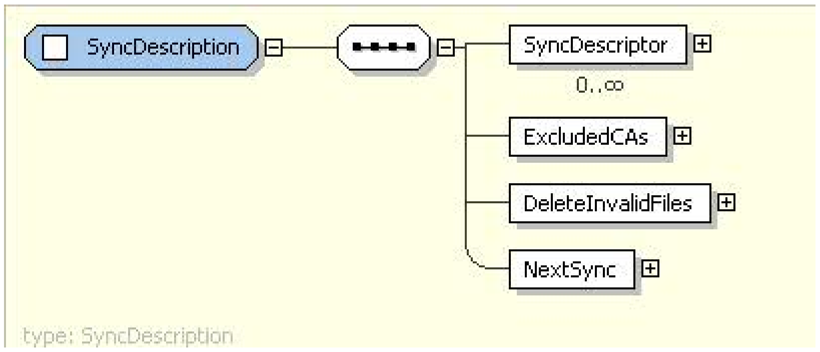


Figure 5-25 SyncGTS: Sync Description

The *SyncDescriptor* element (Figure 5-26) provides details on which GTSs to sync with and the criteria to sync on. The *gtsServiceURI* child element should contain the URI of the GTS to sync with. Each time the SyncGTS, syncs it removes all previously discovered certificate authorities from the trust list unless they are excluded or unless they were discovered earlier and the expiration time attached to the sync has not yet expired. The *Expiration* child element specifies how long certificate authorities discovered during this sync should be valid, providing such a buffer is important in handling unexpected errors such as a short term network outage. The *TrustedAuthorityFilter* child element specifies the criteria that must be met in order for a certificate authority managed by a GTS to be trusted within the context of a local environment.

The following synchronization criteria may be specified per synchronization:

- **Name –** The name of the certificate authority within the GTS.
- **CertificateDN –** The Certificate Authority's subject within its certificate.
- **Trust Level(s) -** A trust level specifies the level of confidence in which a given certificate authority is trusted in the grid. Each certificate authority can be assigned a set of trust levels. The *TrustLevels* element specifies a set of trust levels you require CAs to be assigned in order to trust them.
- **Lifetime –** In a federated trust fabric GTSs inherit certificate authorities from other GTSs. Certificate Authorities are inherited for a specified period of time which expires if not renewed. This element allows you to specify whether or not you will accept those CAs whose lifetime expired.

- **Status -** Specifies the current state of the certificate authority; this allows a certificate authority 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.
- **IsAuthority -** Specifies whether or not the GTS is required to be the authority for candidate certificate authorities.
- **SourceGTS –** Specifies which GTS must be the source of candidate certificate authorities.
- **AuthorityGTS -** Specifies which GTS must be the authority of candidate certificate authorities.

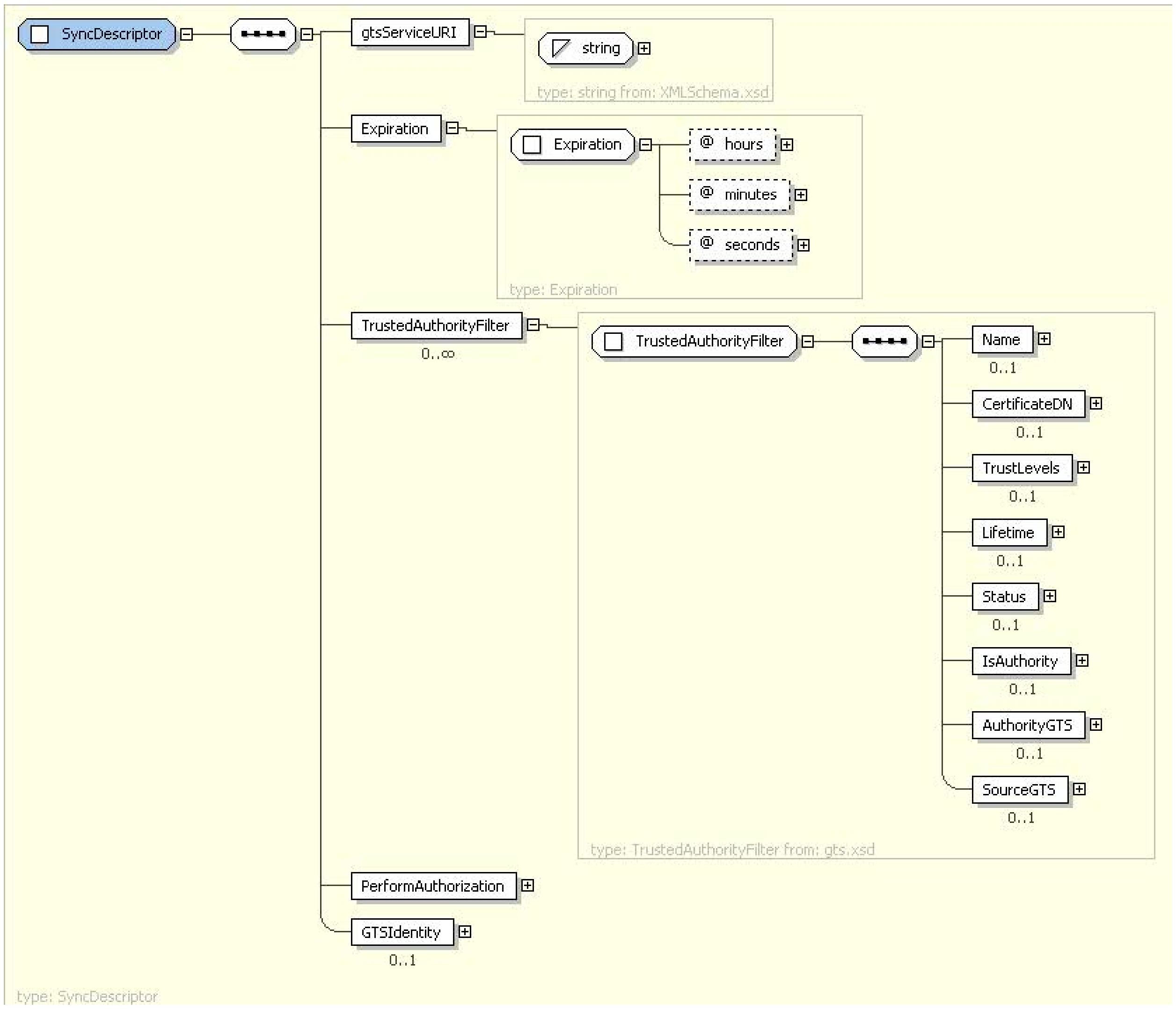


Figure 5-26 SyncGTS: Sync Descriptor

The *PerformAuthorization* child element specifies whether or not to perform authorization against the GTS being synced with, if performing authorization is request, the *GTSIdentity* element should contain the grid identity of the GTS being synced with.

Note that there can be multiple *SyncDescriptor(s)* in a sync description, SyncGTS will process the *SyncDescriptors* in the order that they exist in the document. If a conflict is discovered between *SyncDescriptors,* the information obtained from the *SyncDescriptor* appearing earliest in the document is used. For example, if a certificate authority is discovered twice (through two different *SyncDescriptors)*, the certificate authority's certificate and CRL obtained from the earlier *SyncDescriptor* is included in the trust list.

As mentioned earlier, each time the SyncGTS syncs it removes all previously discovered certificate authorities from the trust list unless they are excluded or unless they were discovered earlier and the expiration time attached to the sync has not yet expired. The *ExcludedCAs* element contains a list of all the CAs (by subject) that should never be removed. CAs listed in the exclude list are generally those that are used for bootstrapping the trust fabric or are outside the trust fabric. SyncGTS also provides the ability to remove any unexpected files that may exist in the Globus trusted certificates directory, the *DeleteInvalidFiles* element allows this to be specified. Finally, the *NextSync* element specifies how often (in seconds) the SyncGTS should sync with the trust fabric. This element is only used if SyncGTS is asked to run constantly, such as the service base approach.

Building SyncGTS

To build the SyncGTS, enter *ant clean all* from the SyncGTS installation directory. Depending on the distribution obtained, you may be required to build from the root distribution directory to ensure that the SyncGTS is provided with all of its dependencies.

Running SyncGTS

The SyncGTS command line utility can be used to sync once with the trust fabric, meaning your environment will authenticate users and services against the trust fabric at the point in time in which you last synced. This command line approach is recommended for clients and is not recommended for servers. To sync once using the command line approach, enter the command *ant syncWithTrustFabric* from the installation directory. Under the service-based approach, the SyncGTS is deployed to a Globus service container and synchronizes with the GTS every X seconds. The service-base approach is recommended for server environments that are constantly inheriting changes in the trust fabric. To deploy the SyncGTS to a service container, enter the following from the SyncGTS installation directory:

```
Deploy to a Globus Container:

ant deployGlobus

Deploy to a Tomcat Container:

ant deployTomcat
```

Grid User 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 X509 Identity Certificates for identifying a user. An X509 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 X509 certificates providing the ability to delegate your credentials to other services, for example 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.

A practical solution to this problem, both from the users' point of view and their institutions is to allow those users to authenticate with the grid through the same mechanism in which they authenticate with their institution. Dorian is a grid user management service that (1) hides the complexities of creating and managing grid credentials from the 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 a complete grid-enabled solution, based on public key certificates and SAML, for managing and federating user identities in a grid environment. Grid technologies have adopted the use of X509 identity certificates to support user authentication. The Security Assertion Markup Language (SAML) has been developed as a standard for exchanging authentication and authorization statements between security domains. Note that grid certificates and SAML assertions serve different purposes. SAML is mainly used between institutions for securely exchanging authentication information coming from trusted identity providers. The primary use of the certificates is to uniquely identify Grid users, facilitate authentication and authorization across multiple resource providers, and enable secure delegation of credentials such that a service or a client program can access resources on behalf of the user. A salient feature of Dorian is that it provides a mechanism for the combined use of both SAML and grid certificates to authenticate users to the grid environment through their institution's authentication mechanism.

One of the challenges in building an identity management and federation infrastructure is to create an architecture that incorporates multiple differing authentication mechanisms used by various institutions. In addressing this challenge two possible approaches are identified. The first is to build an infrastructure that would allow pluggable authentication modules, wherein a module would be developed for each authentication mechanism. In this architecture, a user's authentication information would be routed to the appropriate module that contains the logic for authenticating the user with its institution. Although this approach solves the problem, it requires at least one module be developed for each authentication mechanism. This would require the Grid infrastructure administrators to become intimately familiar with each institution's authentication mechanisms, and would increase the system's complexity with each new module added.

Another approach is for the infrastructure to accept an institutionally supplied, standard "token" as a method of authentication. In this approach, users first authenticate with their institution's identity management system. Upon successfully authentication the institution's identity management system issues a token which can then be given to the federated grid identity management system in exchange for grid credentials. The benefit of this approach over the first is that it does not require writing a plug-in every time a new institutional authentication mechanism comes online. It does, however, require every institutional authentication system to agree upon and be able to provide a common token. As SAML has been adopted by many institutions, the token format was chosen as the basis of the second approach for Dorian.

The Security Assertion Markup Language (SAML) is an XML standard for exchanging authentication and authorization data between security domains. Generally the exchange of authentication and authorization data is made between an *Identity Provider* (IdP) and another party. An institution's authentication system or identity management system is an example of an IdP. Dorian uses SAML authentication assertions as the enabling mechanism for federating users from local institutions to the grid.

Figure 5-27 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 SAML assertion, vouching that the user has authenticated. The user then sends this SAML assertion to Dorian in exchange for grid credentials. Dorian only issues grid credentials to users that supply a SAML assertion from a *Trusted Identity Provider*. Dorian's grid service interface provides mechanisms for managing trusted identity providers; this is discussed in greater detail later in this guide. For example, in Figure 5-27 where a Georgetown user wishes to invoke a grid service that requires grid credentials, they first supply the application with their username and password to the Georgetown Authentication Service as they would normally do. The application client authenticates the Georgetown user with the Georgetown Authentication Service, 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.

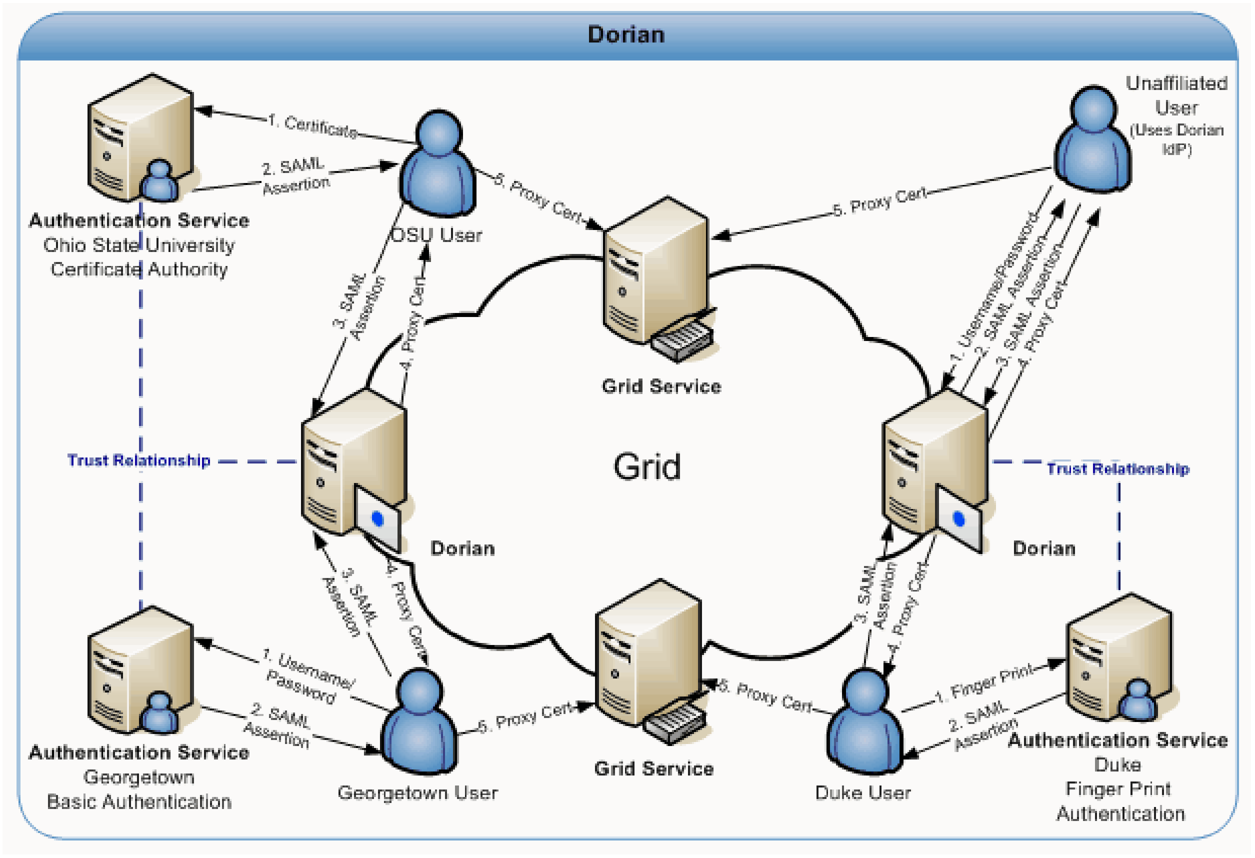


Figure 5-27 Dorian

To facilitate smaller groups or institutions without an existing IdP, Dorian also has its own internal IdP. This allows users to authenticate to Dorian directly, thereby enabling them to access the grid. It provides administrators with facilities for approving and managing users. All of the Dorian IdP's functionality is made available through a grid service interface. Details of the Dorian IdP are provided later in this document. Figure 5-27 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 their 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.

Dorian Software Prerequisites

Table 5-2 lists the software prerequisites for Dorian.

Software	Version	Description
Java SDK	jsdk1.5 or higher	Dorian 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 user accounts and other information.
Ant	Ant 1.6.5	Dorian along with the Globus Toolkit requires Jakarta Ant for building and deploying.
Globus	Globus 4.0.3	Dorian is built on top of the Globus Toolkit. Dorian requires the ws-core installation of the Globus Toolkit.
Tomcat (Only required if deploying to Tomcat)	Tomcat 5.0.30	Dorian can be optionally deployed as a Grid Service to a Tomcat deployed Globus Toolkit.

Table 5-2 Software prerequisites for Dorian

Building Dorian

To build Dorian, enter *ant clean all* from the Dorian installation directory. Depending on the distribution obtained you may be required to build from the root distribution directory to ensure that Dorian is provided with all of its dependencies.

Configuring Dorian

Dorian is configured through a single configuration file which is located at `DORIAN_INSTALLATION_DIRECTORY/etc/dorian-conf.xml`. Dorian uses a Mysql Database as its backend data store; you must provide Dorian with the connection details for your Mysql database (Figure 5-30). The *database* element in the Dorian configuration file is used to specify the connection information for your Mysql database. In the majority of cases you will 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 Dorian is first initialized it creates a database named with the value of the *dorian-internal-id* element. Dorian also proceeds to setup its database schema in the database it created. In order to do so Dorian needs to be configured with a database user that has the appropriate permissions. If you do not wish to provide Dorian with such a user you may create the database manually and provide Dorian with a user who has the permission to modify the database schema. In this scenario Dorian will not create the database but will proceed to setup its database schema in the database that was manually created.

```
<dorian-config>
    <dorian-internal-id>DORIAN</dorian-internal-id>
    <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>
</dorian-config>
```

Figure 5-28

Dorian Identity Provider Configuration

The Dorian Identity Provider is configured in the *idp-config* element, shown in Figure 5-31. The *uid-length* element allows the minimum and maximum user id length to be specified. The *password-length* element allows the minimum and maximum password length to be specified. The *registration-policy* element specifies the registration policy to use. The registration policy tells the Dorian IdP what to do when a new user registers.

The Dorian IdP supports two types of registration policies; manual and automatic. A manual registration policy requires administrators to approve new accounts; an automatic registration process automatically approves new accounts.

- To specify a manual registration policy set the *class* attribute of the *registration-policy* element equal to *gov.nih.nci.cagrid.dorian.service.idp.ManualRegistrationPolicy.*
- To specify an automatic registration policy set the *class* attribute of the *registration-policy* element equal to *gov.nih.nci.cagrid.dorian.service.idp.AutomaticRegistrationPolicy.*

The *asserting-credentials* element specifies the configuration for the Dorian IdP's signing credentials or the credentials Dorian uses in signing SAML Assertions. Currently the DorianIdP automatically generates a set of signing credentials which are signed by the Dorian certificate authority. In the future the Dorian IdP may allow the signing credentials to be imported. With that said the *auto* attribute is required to be set to *true.* The *renew* attribute specifies whether or not the Dorian IdP should renew the signing credentials once they expire. The *key-*

password attribute specifies the password to user in encrypting the private key of the signing credentials; the private key will be stored in Dorian's database.

```
<idp-config>

    <uid-length min="4" max="10"/>

    <password-length min="6" max="10"/>

    <registration-policy class="gov.nih.nci.cagrid.dorian.service.idp.ManualRegistrationPolicy"/>

    <asserting-credentials auto="true" renew="true" key-password="idpkey"/>

</idp-config>
```

Figure 5-31 Dorian IdP Configuration

Dorian Identity Management Configuration

The Identity Management aspects of Dorian are configured in the *ifs-config* element. The *idp-name* element allows the specification of a minimum and maximum IdP name length. The *credential-valid* element specifies how long a user's long term grid credentials are valid for from the time they are created. This lifetime of the grid credentials is limited by the lifetime of the certificate authority, in cases where the lifetime of the grid credentials exceeds the lifetime of the CA, the grid credentials will be issued with the same life time as the CA. The *max-proxy-lifetime* element specifies the maximum amount of time that a user may request a proxy certificate to be created for. The *policies* element specifies a list of user policies that are supported by the Dorian. These policies designate how Dorian should handle users from a specified Identity Provider. 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's 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's 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 administrator is required to grant access. (user's status is pending) When a user's 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 administrator is required to grant access. (user's status is pending) When a user's whose long term certificate expires, an administrator is required to manually renew it.

Besides the four policies released with Dorian, you may also create your own by providing an implementation of the *gov.nih.nci.cagrid.dorian.service.ifs.UserPolicy* class. The *gts-services* element specifies the GTS(s) in which Dorian should publish its certificate authority CRLS. In order for Dorian to publish CRLS to a GTS, the Dorian Certificate Authority must be registered in the GTS as a trusted authority. Dorian must also be granted permission within the GTS to publish a CRL for its certificate authority.

```
<ifs-config>
    <idp-name-length min="4" max="255"/>
    <credentials-valid years="1" months="0" days="0" hours="0" minutes="0" seconds="0"/>
    <max-proxy-lifetime hours="12" minutes="0" seconds="0"/>
    <policies>
     <policy name="Manual Approval / Manual Renewal"
     class="gov.nih.nci.cagrid.dorian.service.ifs.ManualApprovalPolicy"/>
           <policy name="Manual Approval / Auto Renewal"
     class="gov.nih.nci.cagrid.dorian.service.ifs.ManualApprovalAutoRenewalPolicy"/>
     <policy name="Auto Approval / Manual Renewal"
     class="gov.nih.nci.cagrid.dorian.service.ifs.AutoApprovalPolicy"/>
     <policy name="Auto Approval / Auto Renewal"
     class="gov.nih.nci.cagrid.dorian.service.ifs.AutoApprovalAutoRenewalPolicy"/>
    </policies>
   <gts-services>
     <gts-service>https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS</gts-service>
  </gts-services>
</ifs-config>
```

Figure 5-32 Dorian Identity Management Configuration

Dorian Certificate Authority Configuration

Dorian maintains its own certificate authority, which it uses to issue and revoke grid credentials. Depending on your deployment requirements, Dorian can either be configured to use an existing certificate authority or it can generate a new certificate authority. The *dorian-ca-config* element in the Dorian configuration specifies the configuration details for Dorian's certificate authority. The *auto-create* element specifies to Dorian whether or not it should create a new certificate authority. To instruct Dorian to create a new certificate authority set value of the *enabled* element to *true*, if you wish Dorian to use an imported or existing certificate authority set the value of the *enabled* attribute to false. If you have configured Dorian to create a new certificate authority, you must also specify a *time-valid* element, which instruct Dorian how long the new certificate authority should be valid for. You must also specify a Subject DN for the new certificate authority; this can be done in the *ca-subject* element by specifying the Subject DN for the value of the *dn* attribute. It is critical that the DN specified does not conflict with the DN of any other certificate authorities in the grid. Figure 5-33 illustrates an example of a configuration that specifies the creation of a new certificate authority.

```
<dorian-ca-config>
   <ca-password value="admin"/>
   <auto-create enabled="true"/>
      <time-valid years="5" months="0" days="0" hours="0" minutes="0" seconds="0"/>
      <ca-subject dn="O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=localhost,CN= Dorian CA"/>
    </auto-create>
    <auto-renewal enabled="true" years="5" months="0" days="0" hours="0" minutes="0"
seconds="0"/>
</dorian-ca-config>
```

Figure 5-33 Dorian Certificate Authority Configuration (Auto Created)

Figure 5-34 illustrates an example of a configuration that specifies the use of an existing certificate authority. If you have configured Dorian to user an existing certificate authority you must import it before running Dorian for the first time. This can be done from the Dorian installation directory by typing *ant importCA*, doing to will run a command line program that will prompt you for the location of the certificate and private key of the certificate authority.   It should be noted that the import tool requires both the certificate and private key to be in PEM format. You will be also asked to provide a password, if it is needed to decrypt the private key.

```
<dorian-ca-config>
   <ca-password value="admin"/>
   <auto-create enabled="false"/>
   <auto-renewal enabled="true" years="5" months="0" days="0" hours="0" minutes="0" seconds="0"/>
</dorian-ca-config>
```

Figure 5-34 Dorian Certificate Authority Configuration (Imported)

Despite whether your configuration specifies to create a new certificate authority or use an existing one, you should specify a password that Dorian should use for encrypting the CA private key in the database; this is done in the *ca-password* element. It should be noted that the *ca-password* element should be configured before importing a certificate authority, since the import tool will used the ca password specified when encrypting the CA private key. The *auto-renewal* element in the Dorian CA configuration instructs Dorian what to do when a certificate authority expires. If auto renewal is enabled Dorian will renew the certificate authority when it expires for the lifetime specified. If auto renewal is disabled, Dorian will allow the certificate authority to expire, invalidating any grid user accounts that it manages.

Deploying Dorian

Dorian can be deployed to either a HTTPS secure Globus container or an HTTPS secure Tomcat container. It is assumed that both Dorian and the container Dorian is being deployed to have been properly configured prior to deployment. Dorian can be deployed to a Globus container by entering *ant deployGlobus* from the Dorian installation directory. Likewise Dorian can be deployed to a Tomcat container by entering *ant deployTomcat* from the Dorian installation directory.

Setting the 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 will be 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*, we recommend that you change the password immediately; this can be done through the local user management interface of the Dorian IdP, which will be discussed in the *"Managing Dorian IdP Users"* section of this document. 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.

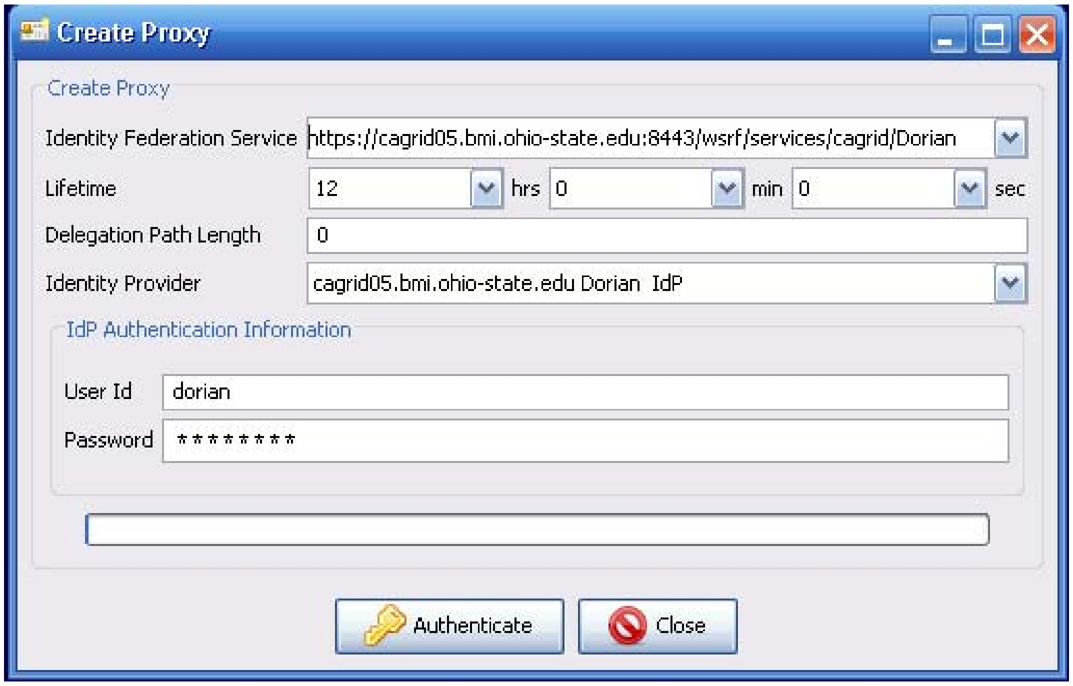
Logging onto the Grid

The GAARDS UI provides the ability to create and manage grid proxies/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 5-35).
2. To login, first specify the Dorian that maintains your grid user account by selecting the Dorian URI from the **Identity Federation Service** drop down. The GAARDS UI is pre-configured with a list of Dorian's through its configuration file. If the Dorian you wish to select is not in the list, enter it.
3. Select the lifetime of your grid proxy; this is how long your credentials are good for. Select from the **Lifetime** drop downs for hours, minutes, and seconds.
4. Specify how many times your credentials can be delegated. Delegating your credentials gives another party the ability to act on you 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. If you wish to increase the delegation path length, change the **Delegation Path Length** text field.
5. Specify the Identity Provider you wish to authenticate with by selecting your Identity Provider from the **Identity Provider** drop down. After selecting your Identity Provider, input fields display requesting the Authentication Information required by the selected

Identity Provider. Provide the information requested. For example in Figure 5-35, 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. Add it by editing the GAARDS UI configuration file.

6. Once you have entered the required IdP Authentication Information, click the **Authenticate** button 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 (Figure 5-36) 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 Proxies.



Identity Federation Service	https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian
Lifetime	12 hrs 0 min 0 sec
Delegation Path Length	0
Identity Provider	cagrid05.bmi.ohio-state.edu Dorian IdP

User Id	dorian
Password	********

Figure 5-35 Dorian: Login / Create Proxy UI

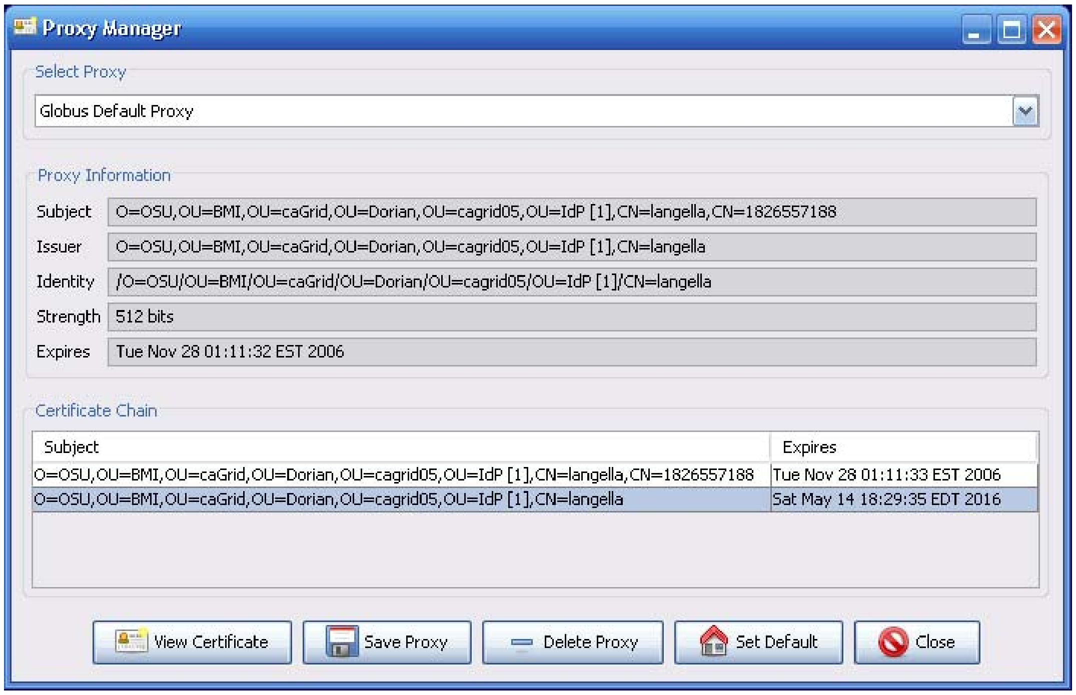
Managing Grid Proxies

The Proxy Manager window (Figure 5-36) allows the management of grid proxies or grid credentials that you have locally created. This window is accessible after logging into Dorian or it is directly accessible through the GAARDS UI as follows.

1. Click the **Grid Account Management** button to open the **Identity Federation** menu. Select **Proxy Management** and click the **Select** button.
2. The **Select Proxy** drop down contains a list of all the non-expired proxies that you created with the addition of the default proxy. Generally Globus clients use the default proxy to connect to grid services if no other proxy is selected. To set the default proxy, select the proxy you wish to make the default from the **Select Proxy** drop down and

click the **Set Default** button. Selecting a proxy from the drop down 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. You can view the details of a certificate in the chain by selecting it and by clicking the **View Certificate** button.

3. Finally, you can delete a proxy by selecting the proxy to delete from the **Select Proxy** drop down. Click the **Delete Proxy** button.



Globus Default Proxy

Proxy Information

Subject: O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cagrid05,OU=IdP [1],CN=langella,CN=1826557188

Issuer: O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cagrid05,OU=IdP [1],CN=langella

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

Strength: 512 bits

Expires: Tue Nov 28 01:11:32 EST 2006

Certificate Chain

Subject	Expires
O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cagrid05,OU=IdP [1],CN=langella,CN=1826557188	Tue Nov 28 01:11:33 EST 2006
O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cagrid05,OU=IdP [1],CN=langella	Sat May 14 18:29:35 EDT 2016

Figure 5-36 Dorian: Proxy Manager UI

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 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's 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's 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 administrator is required to grant access. (user's status is pending) When a user's 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 administrator is required to grant access. (user's status is pending) When a user's 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.

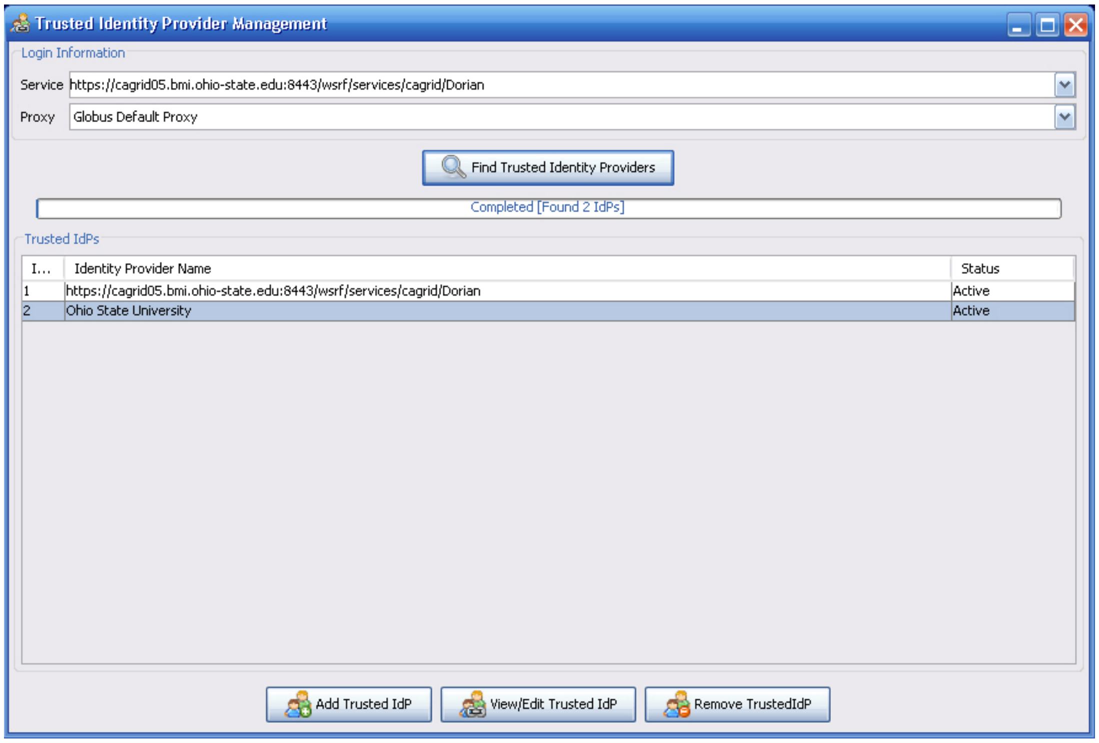
Each Trusted IdP must be configured with a list of acceptable authentication methods. A SAML authentication assertion specifies the method in which the Trusted IdP authenticated the user. In order for the SAML assertion to be accepted by Dorian, the authentication method specified in the assertion must 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. Click the **Grid Account Management** on the GAARDS UI toolbar to open the **Identity Federation** menu. Select **Manage Trusted IdPs** and click the **Select** button.
2. The **Trusted Identity Provider Management** window opens (Figure 5-37). All the IdPs trusted by a Dorian can be listed as follows:
   - From the **Service** drop down, 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.
   - From the **Proxy** drop down, select the proxy or credentials to use to authenticate to Dorian. This must be a proxy of a Dorian administrator.

- Click the **Find Trusted Identity Providers** button. 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 in Figure 5-37, 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 5-37, Dorian would accept credentials from its local IdP and from the Ohio State University.



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

Completed [Found 2 IdPs]

I...	Identity Provider Name	Status
1	https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian	Active
2	Ohio State University	Active

Figure 5-37 Dorian: Trusted IdP Management Admin UI

Adding a Trusted Identity Provider

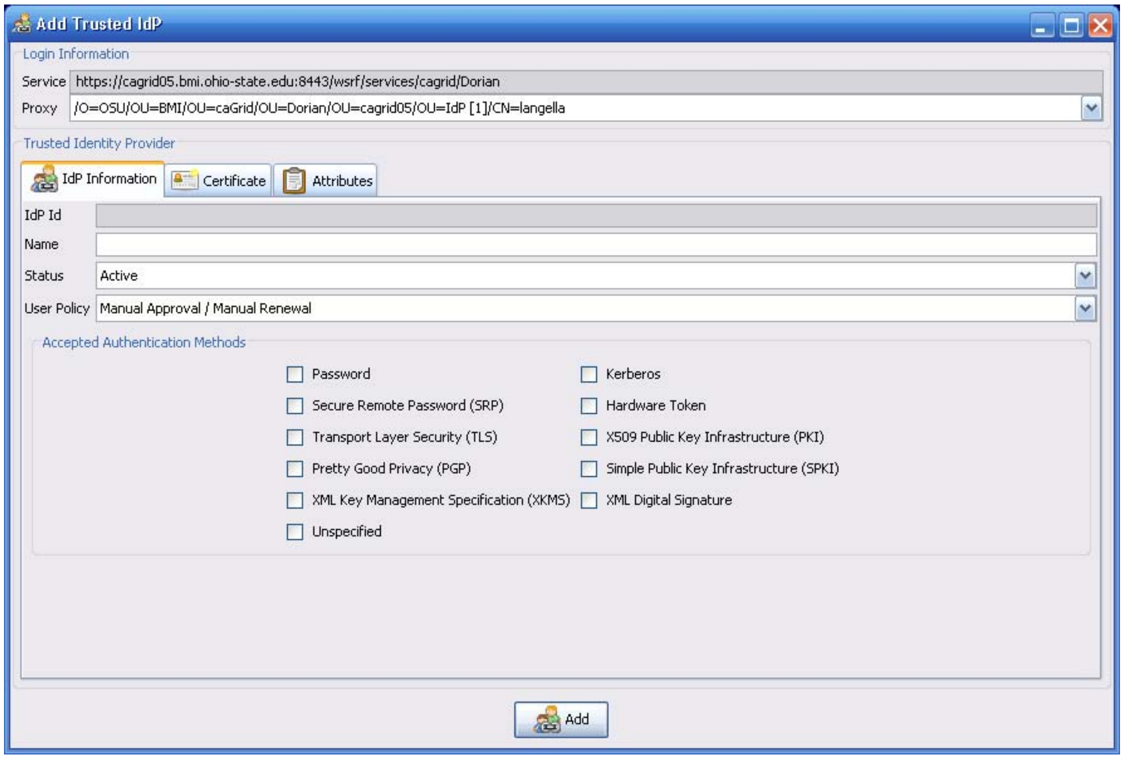
To add a Trusted IdP to Dorian, use the following steps.

1. Click the **Add Trusted IdP** button from the **Trusted Identity Provider Management** window to open the **Add Trusted IdP** window. This (Figure 5-38) consists of three tabs, each of which requires the information to be specified.

   - **IdP Information tab**- specify the name, status, user policy, and acceptable authentication methods.
   - **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** button to open a file browser in which you may browse to the certificate.
   - **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 the **Add** button 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.



Service: https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian
Proxy: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella
IdP Id:
Name:
Status: Active
User Policy: Manual Approval / Manual Renewal

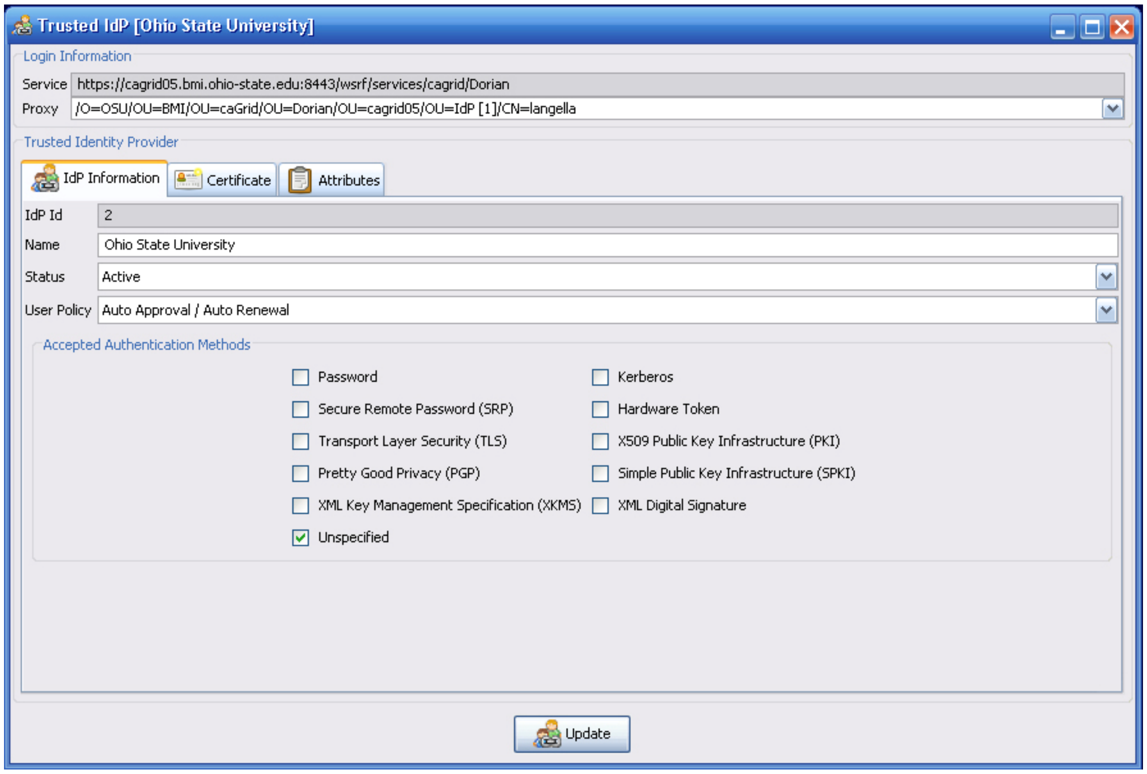
☐ Password
☐ Kerberos
☐ Secure Remote Password (SRP)
☐ Hardware Token
☐ Transport Layer Security (TLS)
☐ X509 Public Key Infrastructure (PKI)
☐ Pretty Good Privacy (PGP)
☐ Simple Public Key Infrastructure (SPKI)
☐ XML Key Management Specification (XKMS)
☐ XML Digital Signature
☐ Unspecified

Figure 5-38 Dorian: Adding a Trusted Identity Provider Admin UI

Viewing/Updating a Trusted Identity Provider

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

1. Select the Trusted IdP of interest and click the **View/Edit Trusted IdP** button from the **Trusted Identity Provider Management** window to open the **Trusted IdP** window. The **Trusted IdP** window (Figure 5-39) consists of three tabs:
   - **IdP Information tab -** update the name, status, user policy, and acceptable authentication methods.
   - **Certificate tab** - update 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 the **Import Certificate** button to open a file browser in which you may browse to the certificate.
   - **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 the **Update** button to commit the changes, which take effect immediately.


Trusted IdP [Ohio State University]
Login Information
Trusted Identity Provider
IdP Information
Certificate
Attributes
Accepted Authentication Methods
Update

Service: https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian
Proxy: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella

IdP Id: 2
Name: Ohio State University
Status: Active
User Policy: Auto Approval / Auto Renewal

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

Figure 5-39 Dorian: View/Update Trusted IdP Admin UI

Removing a Trusted Identity Provider

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

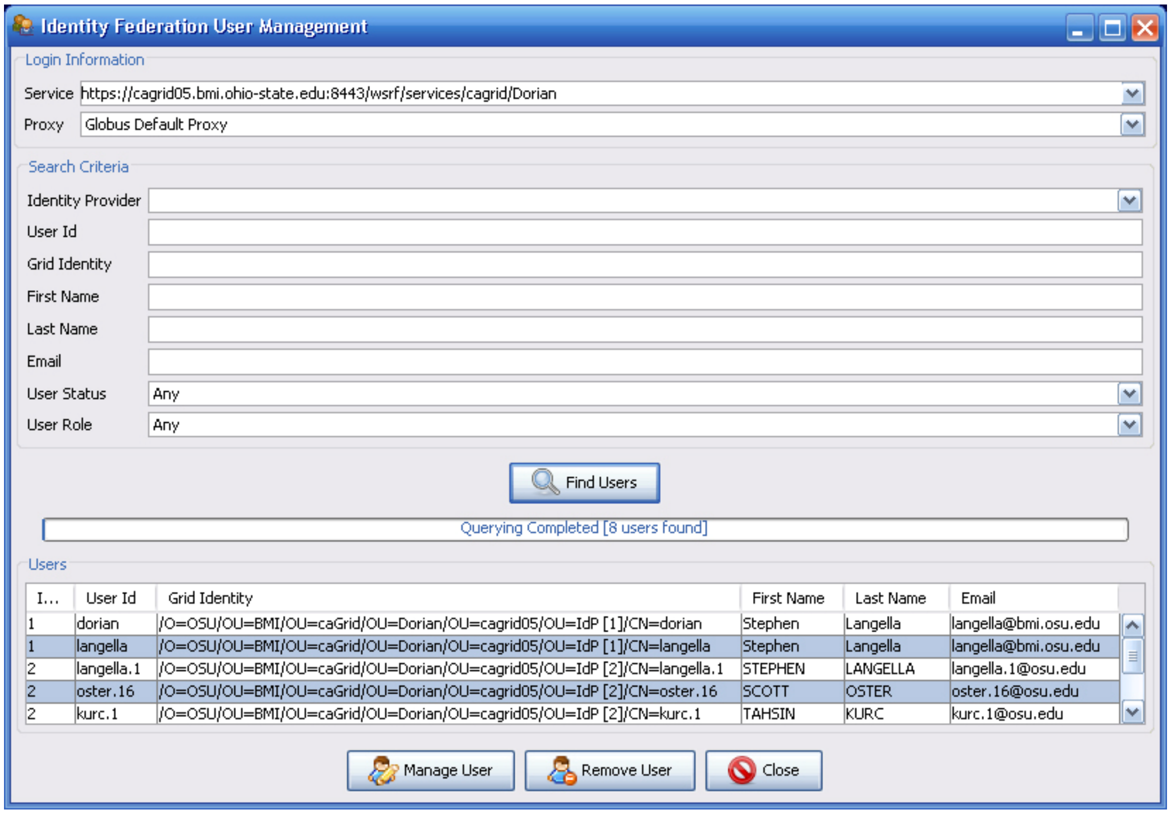
Grid User Account Management

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 a status associated:

Active, Suspended, Pending, or Expired. Only users with an *Active* status will be 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 will be *Active*. When a user's long term certificate expires, the status of the user's account will be 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 will automatically renew the user's long term certificate and private key, and the user's account status will remain *Active*. As mentioned earlier, users who 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. User with the *User* role may only create grid proxies.

Use the following steps to administrate grid user accounts using the GAARDS UI.

1. Click the **Grid Account Management** button on the GAARDS UI toolbar to open the **Identity Federation** menu. Select **User Management** and click the **Select** button.
2. The **Identity Federation User Management** window also known as the Grid User Management window opens (Figure 5-40). 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. 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. Select a proxy of a Dorian administrator.
4. Finally, click the **Find Users** button 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, status, user status, and user role. For example if you wish to search for all the accounts that are pending administrative approval, select **Pending** from the **User Status** drop down.



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

Identity Provider	
User Id	
Grid Identity	
First Name	
Last Name	
Email	
User Status	Any
User Role	Any

Querying Completed [8 users found]

I...	User Id	Grid Identity	First Name	Last Name	Email
1	dorian	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=dorian	Stephen	Langella	langella@bmi.osu.edu
1	langella	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [1]/CN=langella	Stephen	Langella	langella@bmi.osu.edu
2	langella.1	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1	STEPHEN	LANGELLA	langella.1@osu.edu
2	oster.16	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=oster.16	SCOTT	OSTER	oster.16@osu.edu
2	kurc.1	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=kurc.1	TAHSIN	KURC	kurc.1@osu.edu

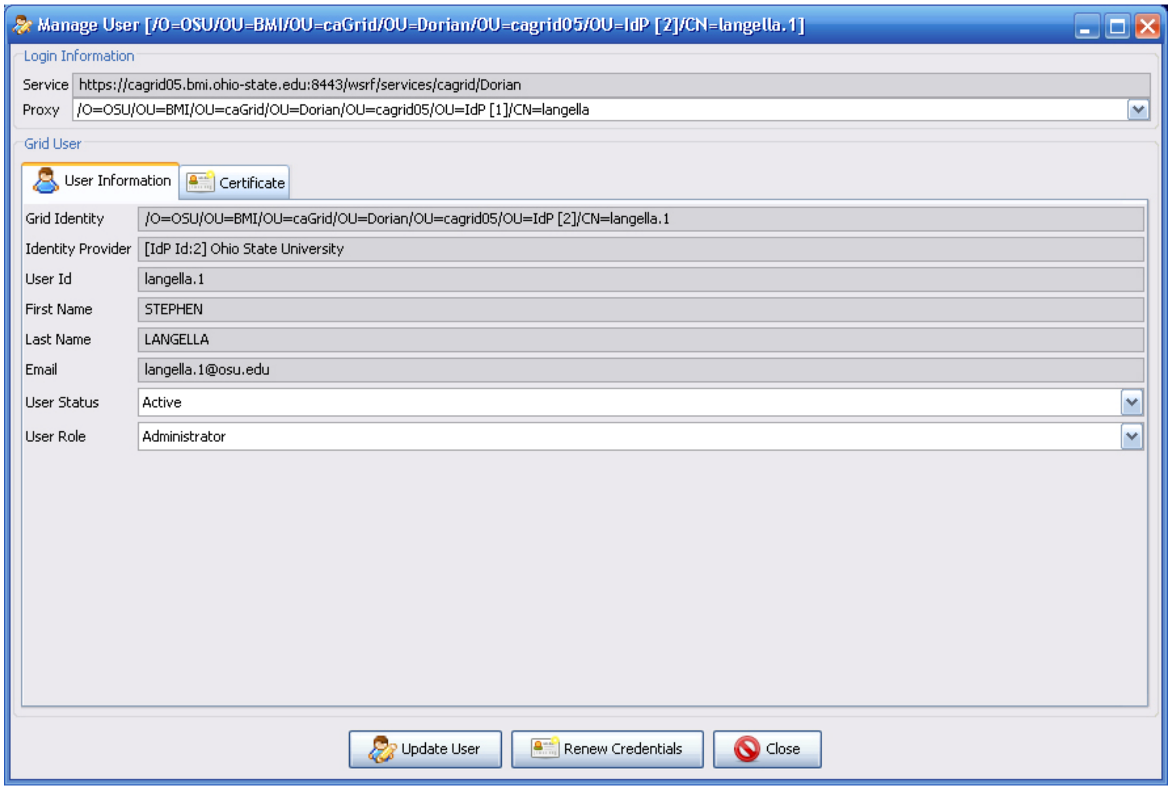
Figure 5-40 Dorian: Grid User Management Admin UI

User Account Management

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

1. Open the **Identity Federation User Management** window by first selecting the user of interest and then clicking the **Manage User** button.
2. The **Manage User** window opens (Figure 5-41). From this window, you can 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*. You may also change the user's role within Dorian. To grant administrative rights, change the user's role from *User* to *Administrator.* To revoke a user's administrative rights, change the user's role from *Administrator* to *User.*
3. To commit changes made to a user's status or a user's role, click the **Update User** button, 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 the **Renew Credentials** button.


Manage User [/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1]

Login Information

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

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

Grid User

Grid Identity	/O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1
Identity Provider	[IdP Id:2] Ohio State University
User Id	langella.1
First Name	STEPHEN
Last Name	LANGELLA
Email	langella.1@osu.edu
User Status	Active
User Role	Administrator

Figure 5-41 Dorian: Manage Grid User Account Admin UI

Removing a Grid User Account

To remove individual grid user accounts through the GAARDS UI, do the following.

Open the **Identity Federation User Management** window by first selecting the user to remove and then clicking the **Remove User** button. It is important to 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.

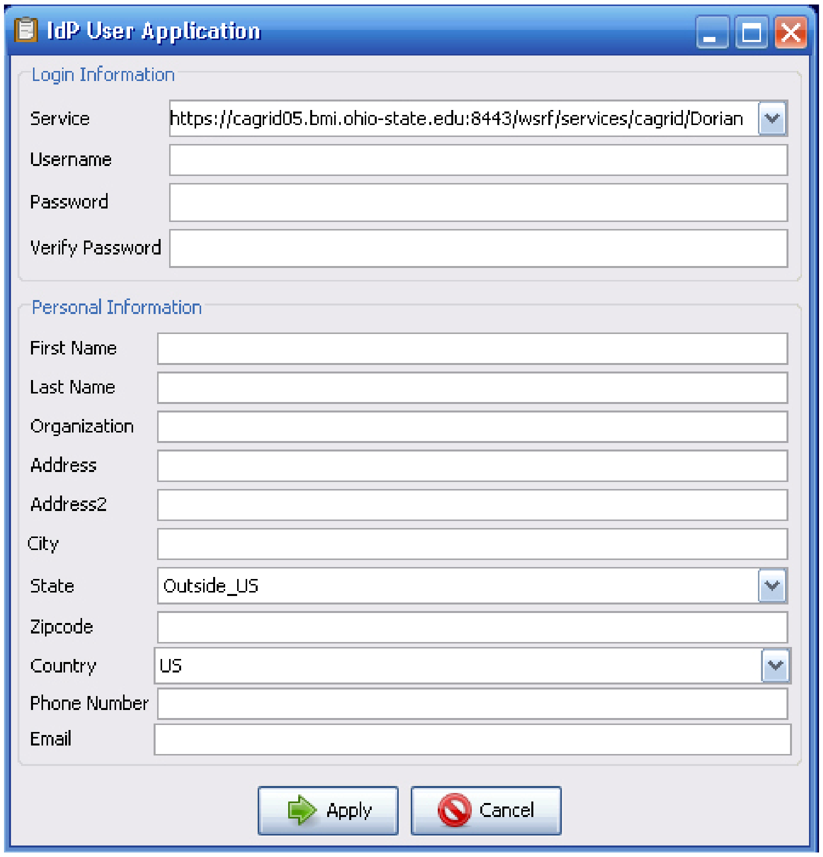
Local Dorian Identity Provider

It is anticipated that most users will use their existing locally provided credentials for obtaining grid credentials and only users that are un-affiliated 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 provides a method for perspective users to register with the Dorian IdP. The GAARDS UI also provides a mechanism for Dorian IdP administrators to administrate Dorian IdP user accounts.

Registering with the Dorian IdP

To register with the DorianIdP through the GAARDS UI, use the following steps.

1. Click the **Local Account Management** button on the GAARDS UI toolbar to open the **Identity Provider** menu. Select **Register** and click the **Select** button to open the **IdP User Application** (Figure 5-42).
2. To register first 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.
3. Finally enter your personal information and click the **Apply** button. 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.



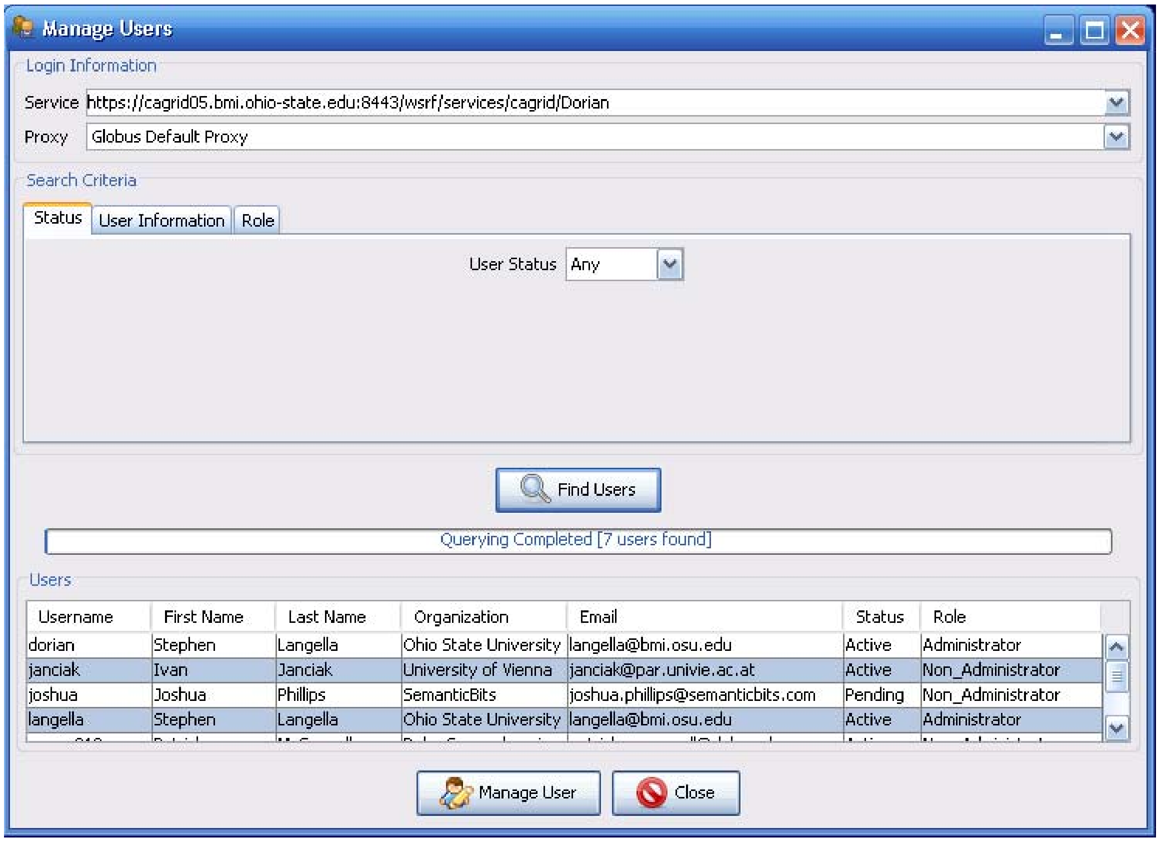
Service	https://cagrid05.bmi.ohio-state.edu:8443/wsrf/services/cagrid/Dorian
Username	
Password	
Verify Password	
First Name	
Last Name	
Organization	
Address	
Address2	
City	
State	Outside_US
Zipcode	
Country	US
Phone Number	
Email	

Figure 5-42 Dorian: Dorian IdP Registration UI

Managing Dorian IdP Users

To manage Dorian IdP users through the GAARDS UI, use the following steps.

1. Click the **Local Account Management** button on the GAARDS UI toolbar to open the **Identity Provider** menu. Select **Manage Users** and click the **Select** button to open the **Manage Users** window (Figure 5-43).
2. From the **Manage Users** window, you may search for local user accounts managed by the Dorian IdP, manage user accounts, and remove user accounts. 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. If you don't see the URI of the Dorian you are interested in, enter it.
3. Next select the grid proxy to use from the **Proxy** drop down. You will need to select a proxy of a Dorian IdP administrator.
4. Finally click the **Find Users** button to list all the local user accounts managed by the selected Dorian IdP. To narrow your search, you may also 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.



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

Proxy: Globus Default Proxy

User Status: Any

Querying Completed [7 users found]

Username	First Name	Last Name	Organization	Email	Status	Role
dorian	Stephen	Langella	Ohio State University	langella@bmi.osu.edu	Active	Administrator
janciak	Ivan	Janciak	University of Vienna	janciak@par.univie.ac.at	Active	Non_Administrator
joshua	Joshua	Phillips	SemanticBits	joshua.phillips@semanticbits.com	Pending	Non_Administrator
langella	Stephen	Langella	Ohio State University	langella@bmi.osu.edu	Active	Administrator

Figure 5-43 Dorian: Dorian IdP User Management UI

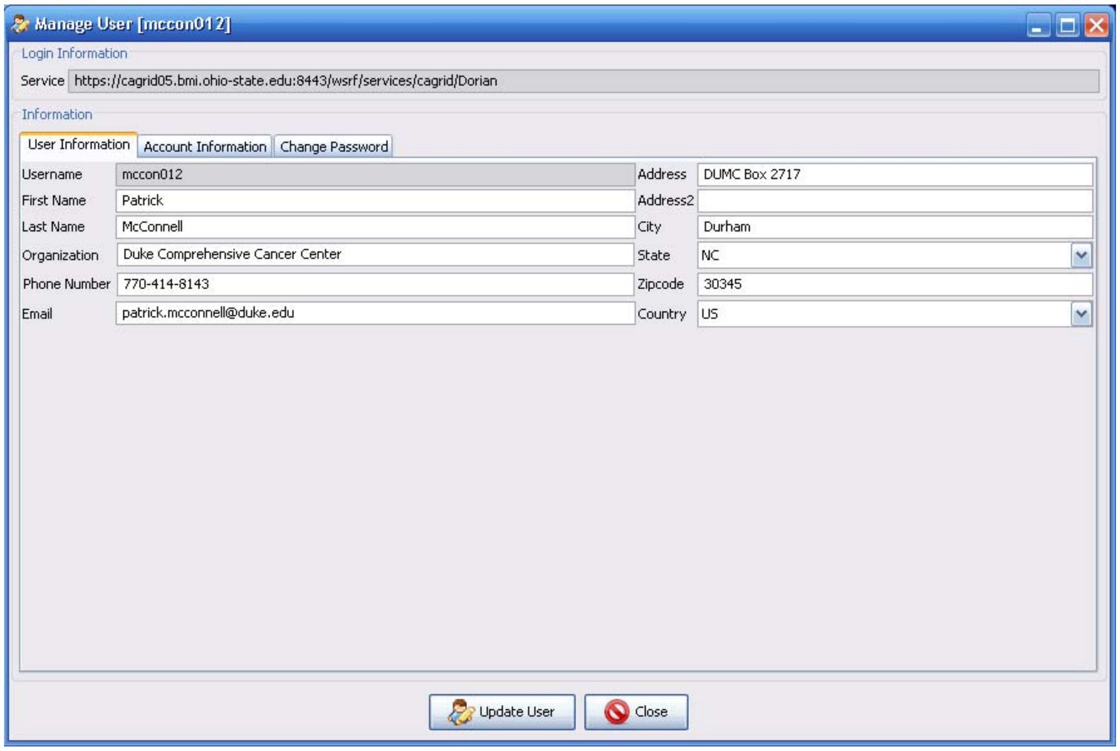
To manage individual local user accounts through the GAARDS UI, use the following steps.

1. Open the **Manage Users** window by first selecting the user of interest and then clicking the **Manage User** button to open the **Manage User** window (Figure 5-44).
2. From the **Manage User** window, 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 consist 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 user's 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.* It is important to note that 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 will depend on the user policy configure 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; this can be done 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	mccon012	Address	DUMC Box 2717
First Name	Patrick	Address2	
Last Name	McConnell	City	Durham
Organization	Duke Comprehensive Cancer Center	State	NC
Phone Number	770-414-8143	Zipcode	30345
Email	patrick.mcconnell@duke.edu	Country	US

Figure 5-44 Dorian: Dorian IdP Manage User UI

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 5-45) 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.

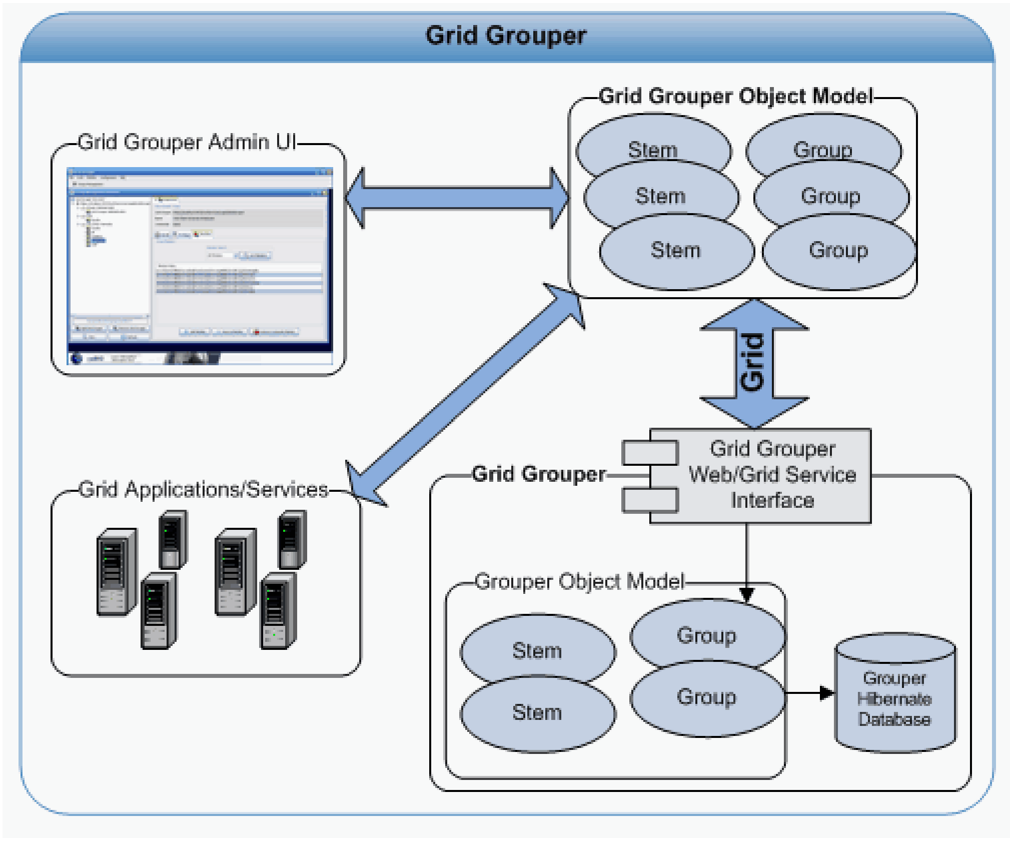

Grid Grouper
Grid Grouper Admin UI
Grid Grouper Object Model
Stem
Stem
Stem
Group
Group
Group
Grid
Grid Applications/Services
Grid Grouper
Grid Grouper Web/Grid Service Interface
Grouper Object Model
Stem
Stem
Group
Group
Grouper Hibernate Database

Figure 5-45 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 for each Faculty, Staff, and Students.

Grid Grouper Software Prerequisites

Table 5-3 lists the software prerequisites for Grid Grouper.

Software	**Version**	**Description**
Java SDK	jsdk1.5 or higher	Grid Grouper is written in Java therefore it requires

Software	Version	Description
		the Java SDK. After installing you will have to 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	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 5.0.30	GridGrouper can be optionally deployed as a Grid Service to a Tomcat deployed Globus Toolkit.

Table 5-3 Software prerequisites for Grid Grouper

Building Grid Grouper

To build GridGrouper, enter *ant clean all* from the GridGrouper installation directory. Depending on the distribution obtained you may be required to build from the root distribution directory to ensure that GridGrouper is provided with all of its dependencies.

Configuring Grid Grouper

To configure GridGouper you must specify your Mysql database information in the `grouper.hibrenate.properties` configuration file located in `GRID_GROUPER_INSTALLATION_DIRECTORY/resources/conf/`.

The properties you need to edit are highlighted in bold in Figure 5-46, 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        =
```

Figure 5-46 Grid Grouper Configuration File

Once you have edited the Grid Grouper configuration file, initialize the Grid Grouper database by manually creating the grouper database in Mysql. The database should be name as configured in the `hibernate.connection.url` property of the `grouper.hibrenate.properties` configuration file. Once you have created the database, enter *ant grouperInit* to build out and initialize the Grouper/Grid Grouper database.

Deploying Grid Grouper

Grid Grouper can be deployed to either an HTTPS secure Globus container or an HTTPS secure Tomcat container. It is assumed that both Grid Grouper and the container it is being deployed to will be properly configured prior to deployment. Grid Grouper can be deployed to a Globus container by entering *ant deployGlobus* from the Grid Grouper installation directory. Likewise Grid Grouper can be deployed to a Tomcat container by entering *ant deployTomcat* from the Grid Grouper installation directory. It is important to note that you must add an initial administrator to Grid Grouper before starting your container. Refer to the next section for further directions.

Grid Grouper Administration

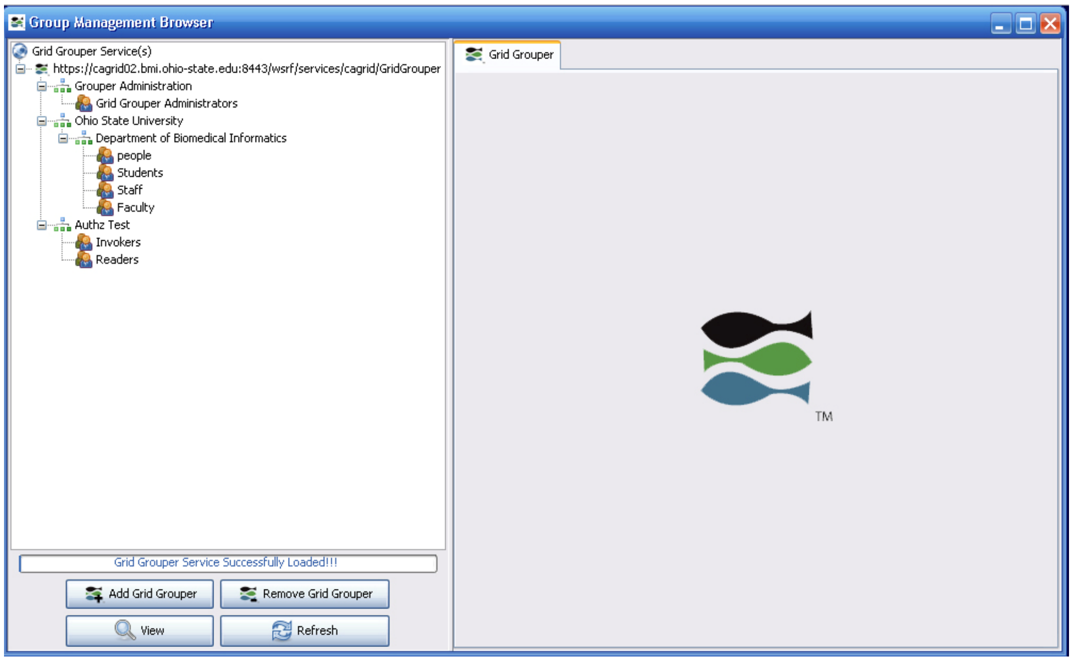
Initially Grid Grouper has a root stem with one child stem named *Grouper Administration* (grouperadministration). The Grouper Administrative stem contains one group named *Grid Grouper Administrators* (grouperadministration:gridgrouperadministrators). The *Grid Grouper Administrators* is the super user group for Grid Grouper; all members of this group have admin privileges on all the stems and groups within Grid Grouper. It is important to note the individual groups and stems can also be assigned administrators.

The *Grid Grouper Administrators* group is initially empty, but at least one administrative user must be added before Grid Grouper can be administered. Grid Grouper provides a command line tool for bootstrapping GridGrouper and initially adding administrator(s). The command line tool for adding administrators can be invoked by entering *ant addAdmin* from the Grid Grouper distribution directory. The program prompts for the grid identity of the administrator to add. Enter it and press the ENTER key to add the requested user to Grid Grouper as an administrator. After the initial administrator is added, the GAARDS UI should be used for adding additional administrators by adding the user's grid identity as a member of the *Grid Grouper Administrators* group. Instructions on adding members to groups is described in Group Memberships on page 120.

To both browse and administer Grid Grouper through the GAARDS UI, use the following steps.

1. Open the **Group Management Browser** by clicking the **Group Management** button on the toolbar (Figure 5-47). The **Group Management Browser** is divided into two sections. The left is a hierarchal view of the stem/group organization for Grid Grouper(s); the right is a detailed tabbed pane for individually administering both stems and groups. Double clicking on a stem or group from the hierarchal view (left side) opens a tab on the right containing a detailed view of the stem or group and allows the stem or group to be administered.

2. After opening, the hierarchal view in the Group Management Browser does not have any Grid Grouper(s) displayed. To add a Grid Grouper, click the **Add Grid Grouper** button to open the **Add Grid Grouper** dialog. From this dialog, select the URI of the Grid Grouper to be loaded. If the URI for the Grid Grouper to be added is not in the list, enter it.
3. Select the grid proxy to connect to Grid Grouper. You do not need to specify a proxy; Grid Grouper only allows access to things you have permission to do based on the grid proxy you provide it. If you do not provide a grid proxy then you will only be able to access publicly available things. Likewise you will be able to access anything if you provide a grid proxy of a user that is a member of the *Grid Grouper Administrators* group.
4. Once you have specified or not specified a proxy, click the **Add** button to load the specified Grid Grouper into the Group Management Browser.



Grid Grouper Service Successfully Loaded!!!

Figure 5-47 Grid Grouper: Group Management Browser

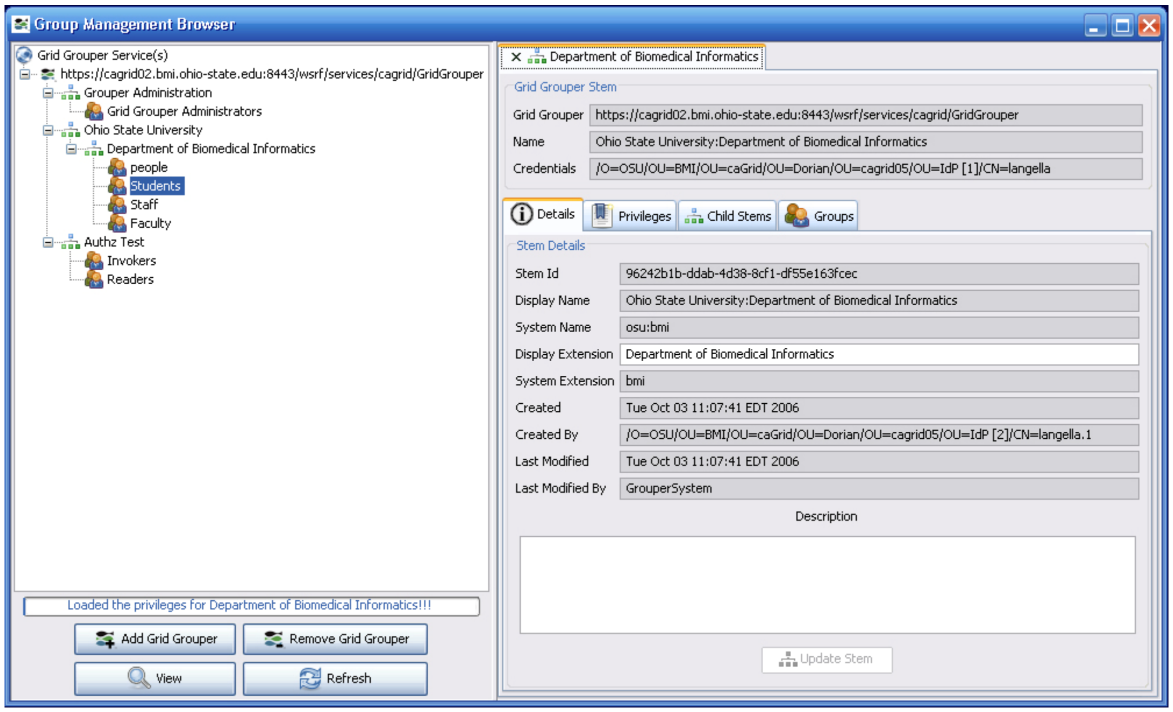
Administrating Stems

To administer a stem, use the following steps.

1. From the Grid Grouper hierarchy in the **Group Management Browser**, select the stem you want to administer and click the **View** button. A tab opens entitled with the stem's name in the Details pane (Figure 5-48). 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.

2. Of the metadata listed, only the display extension and description may be updated by making changes and clicking the **Update Stem** button.



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

Loaded 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

Stem Details

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

Description

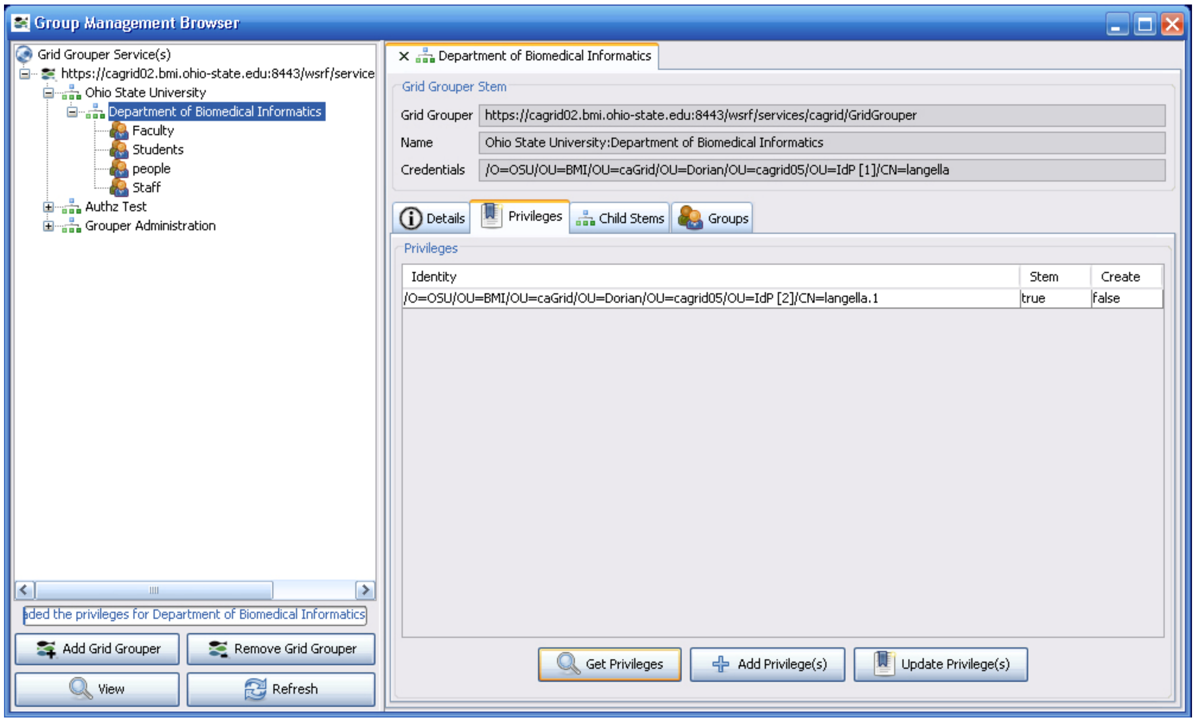
Figure 5-48 Grid Grouper: Administrating Stems UI

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.

To administer stem privileges from the GAARDS UI, use the following steps.

1. From the Grid Grouper hierarchy in the **Group Management Browser,** select the stem whose privileges you want to administer and click the **View** button. The **Stem Administration** tab opens for the selected stem.
2. Next select the **Privileges** tab (Figure 5-49). To list all the privileges for a stem, click the **Get Privileges** button. All the users with privileges on the stem are listed and the privileges that each user has. For example in Figure 5-49, the stem shown lists one user that has been assigned privilege(s). The user listed has been assigned the *Stem* privilege.



ded 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: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

Figure 5-49 Grid Grouper: Administrating Stem Privileges UI

3. Users without existing privileges can be granted privileges by clicking the **Add Privilege(s)** button. For users with existing privileges, new privileges can be added or existing privileges can be revoked, by selecting the user from the **Privileges** *table* (Figure 5-49) and clicking the **Update Privilege(s)** button. In either case, the **Update Stem Privilege(s)** window (Figure 5-50) opens. If you are granting privileges to a user without existing privileges, you will need to specify the user's grid identity and select the privileges you wish to grant them. If you are granting/revoking privileges to/from a user with existing privileges you need to select the privileges you wish to grant and deselect the privileges you wish to revoke.
4. To commit any changes to Grid Grouper, click the **Update Privilege(s)** button. Changes are effective immediately.

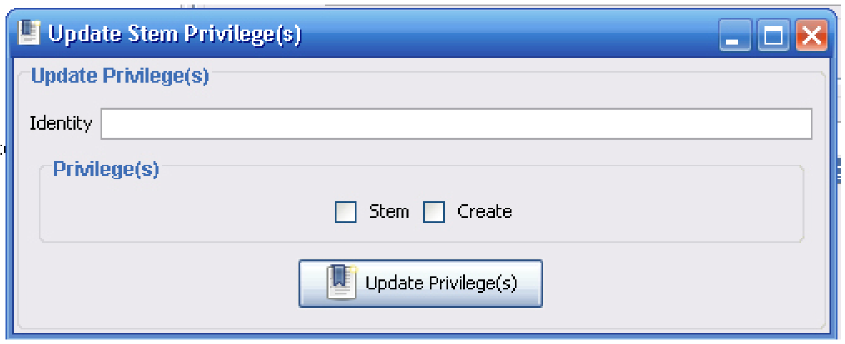
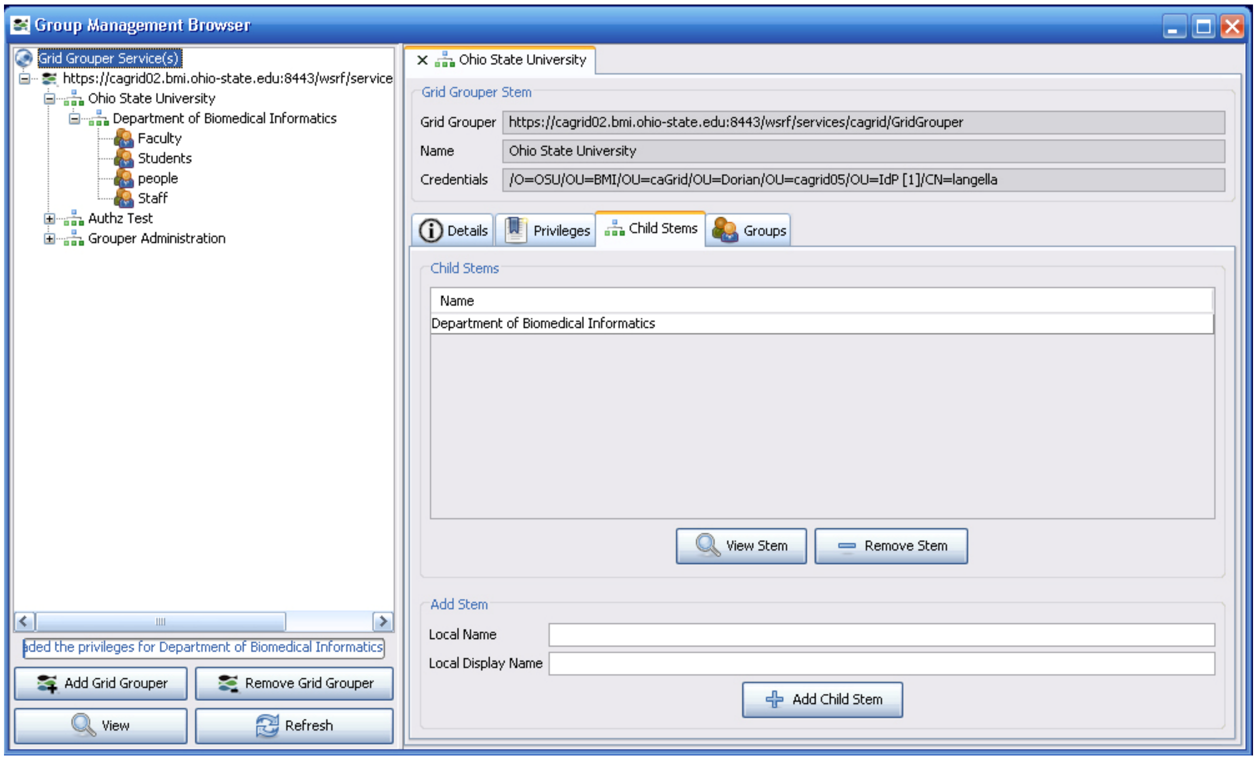


Figure 5-50 Grid Grouper: Granting/Revoking Stem Privileges UI

Managing Child Stems

Each stem in Grid Grouper can have a set of child stems. To list, create, and remove child stems using the GAARDS UI, use the following steps.

1. 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, select the stem from the Grid Grouper hierarchy in the **Group Management Browser** and click the **View** button. The **Stem Administration** tab opens for the selected stem.
2. Next select the **Child Stems** tab. The stem's child stems are listed in the **Child Stems** table (Figure 5-51). To view a child stem, select the stem of interest and click the **View Stem** button to open the **Stem Administration** tab for the selected stem. To remove a stem, select the stem to remove and click the **Remove Stem** button. To remove a stem, all of the stem's child stem(s) and groups must be removed. A child stem can be added at the bottom of the ***Child Stems*** tab by entering a local name for the stem, entering a local display name for the stem, and clicking the **Add Stem** button.



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

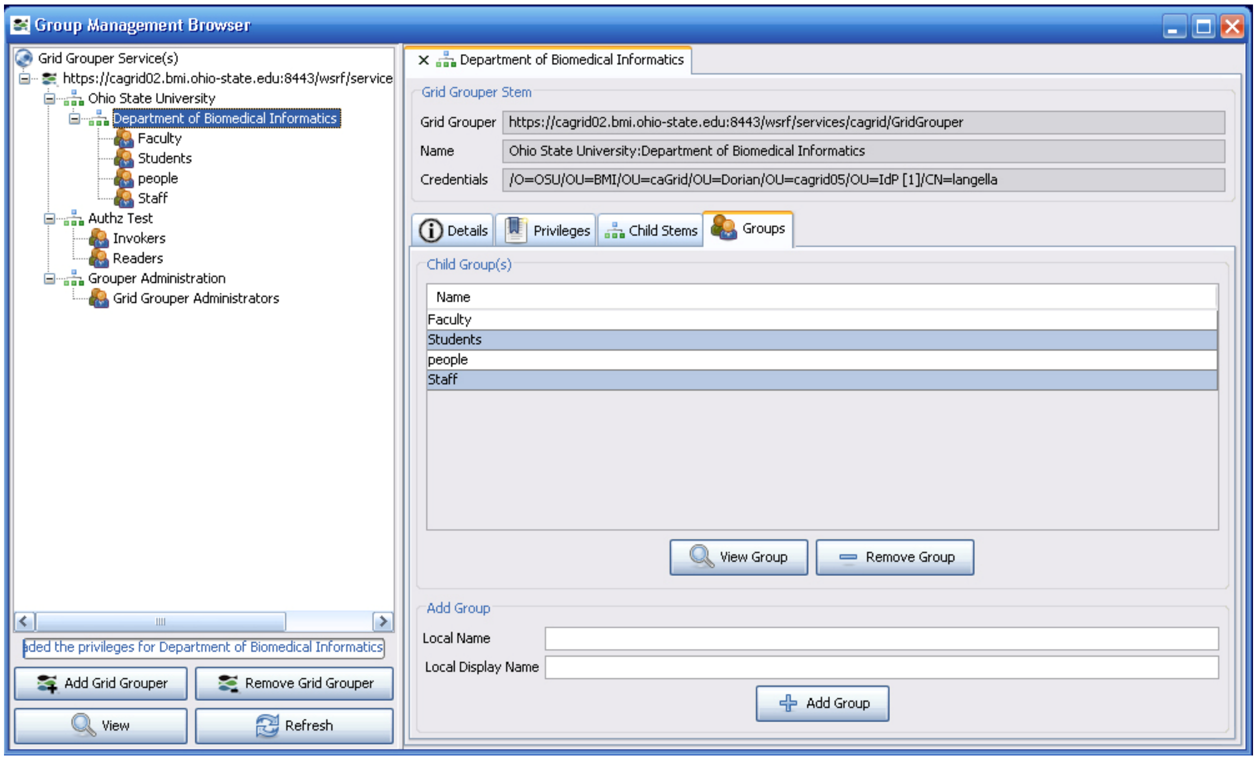
Name
Department of Biomedical Informatics

Figure 5-51 Grid Grouper: Administrating Child Stems UI

Managing Child Groups

Each stem in Grid Grouper can have a set of groups. To list, create, and remove groups through the GAARDS UI, use the following steps.

1. Only users with the *Create Privilege* may create and remove groups. To view the groups for a given stem, select the stem from the Grid Grouper hierarchy in the **Group Management Browser** and click the **View** button. The **Stem Administration** tab for the selected stem opens.
2. Next select the **Groups** tab to list the stem's groups in the **Child Group(s)** table (Figure 5-52). To view a group, select the stem of interest, click the **View Group** button to open the **Group Administration** tab for the selected group. To remove a group, select the group to remove and click the **Remove Group** button. To add a new group, at the bottom of the **Groups** tab, enter a local name for the group, enter a local display name for the group, and click the **Add Group** button.



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

Name
Faculty
Students
people
Staff

Figure 5-52 Grid Grouper: Managing Child Groups UI

Administrating Groups

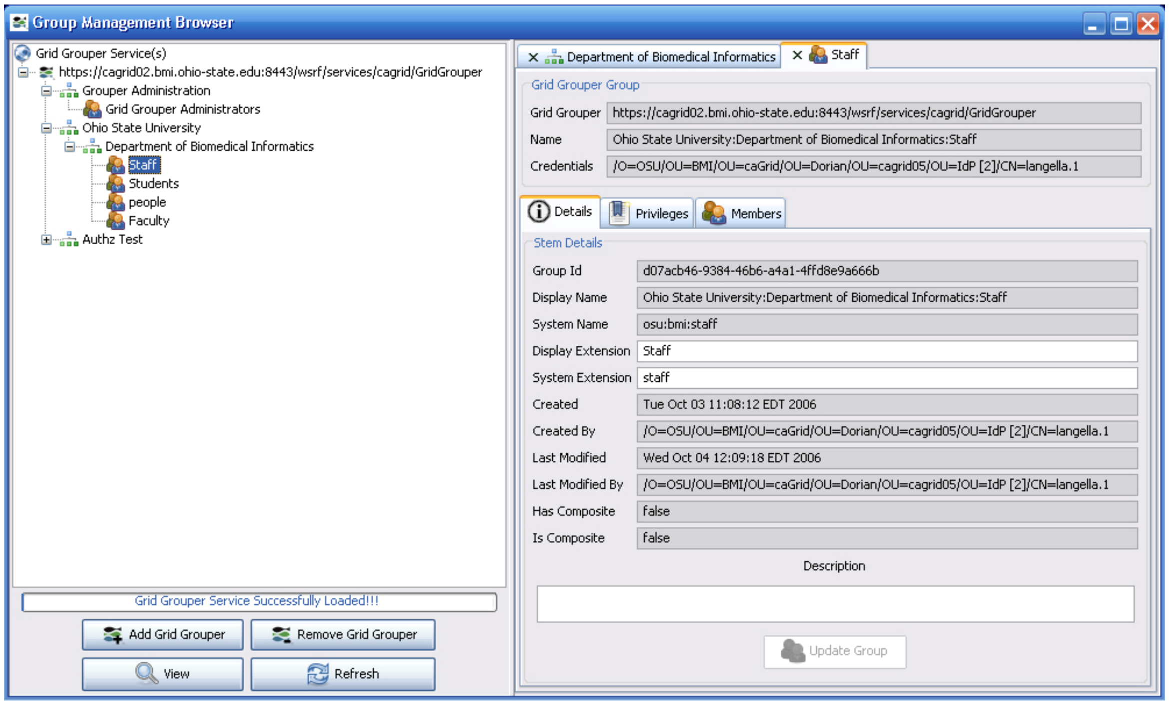
In Grouper/Grid Grouper, groups are comprised of a set of metadata describing the group, a set of members in the groups, and a set of privileges assigned to users for protecting access to the group. To administrate a group, use the following steps.

1. From the Grid Grouper hierarchy in the **Group Management Browser**, select the group to administer and click the **View** button. A tab, entitled with the group's name in the details pane, opens (Figure 5-53). 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.


Group Management Browser
Grid Grouper Service(s)
https://cagrid02.bmi.ohio-state.edu:8443/wsrf/services/cagrid/GridGrouper
Grouper Administration
Grid Grouper Administrators
Ohio State University
Department of Biomedical Informatics
Staff
Students
people
Faculty
Authz Test
Grid Grouper Service Successfully Loaded!!!
Add Grid Grouper
Remove Grid Grouper
View
Refresh
Department of Biomedical Informatics
Staff
Details
Privileges
Members
Update Group

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 5-53 Grid Grouper: Group Management Details UI

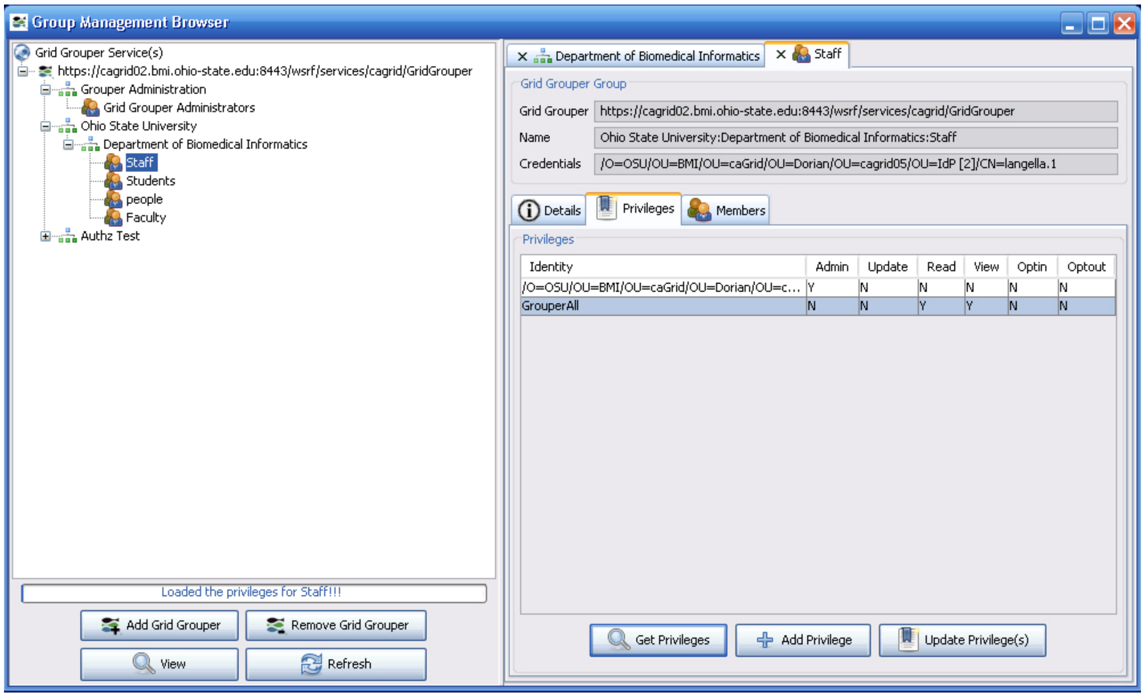
Group Privileges

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*. The *View* privilege allows user's to 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 *Optout* privilege can 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 will 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 administer group privileges from the GAARDS UI, use the following steps.

1. From the Grid Grouper hierarchy in the **Group Management Browser,** select the group whose privileges you want to administer and click the **View** button. The **Group Administration** tab for the selected group opens.
2. Next select the **Privileges** tab (Figure 5-54). To list all the privileges for a group, click

the **Get Privileges** button. All the users with privileges on the group are listed with the privileges each user has. For example in Figure 5-54, the group shown lists two users that have been assigned privilege(s). The first user listed has been assigned the *Admin* privilege; the second user, *GrouperAll,* has been assigned *Read* and *Write* privileges.



Loaded the privileges for Staff!!!

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 5-54 Grid Grouper: Administering Group Privileges UI

3. Users without existing privileges can be granted privileges by clicking the **Add Privilege(s)** button. For users with existing privileges, new privileges can be added or existing privileges can be revoked by selecting the user from the **Privileges** table and clicking the **Update Privilege(s)** button. In either case, the **Update Group Privilege(s)** window opens (Figure 5-55). If you are granting privileges to a user without existing privileges, specify the user’s grid identity and select the privileges to grant them. If you are granting/revoking privileges to/from a user with existing privileges, select the privileges to grant and deselect the privileges to revoke.
4. To commit changes to Grid Grouper, click the **Update Privilege(s)** button. Changes are effective immediately.

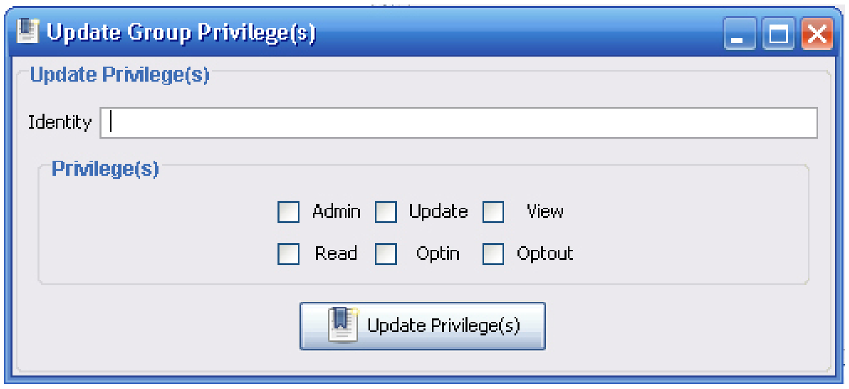


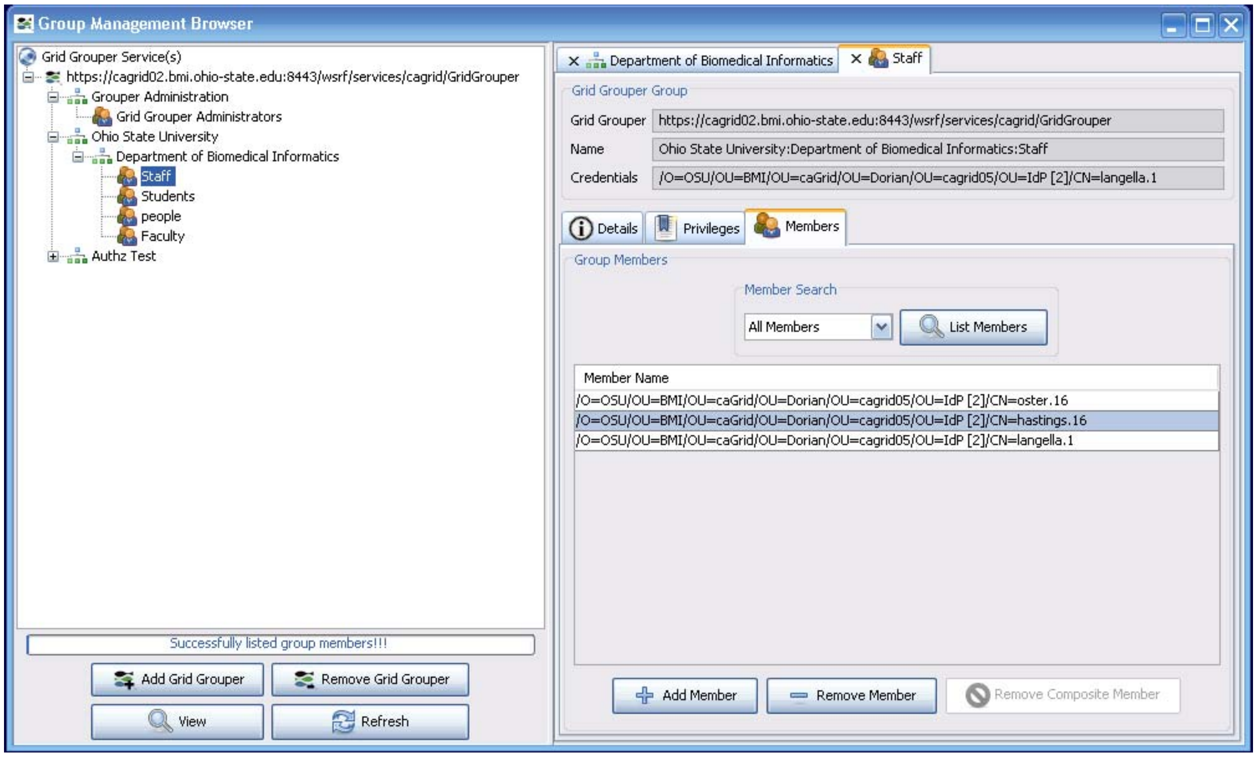
Figure 5-55 Grid Grouper: Updating Group Privileges UI

Group Memberships

Grid Grouper provides three mechanisms for adding members to a group: 1) directly adding a member, 2) adding a subgroup to a group, and 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 sub group are referred to as *Effective Members*. A group can also be set to be a Composite group, which is a group whose memberships are determined 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*.

To administer group memberships through the GARRDS UI, use the following steps.

1. Select the group of interest from the Grid Grouper hierarchy in the **Group Management Browser** and click the **View** button. The **Group Administration** tab opens for the selected group.
2. Next select the **Members** tab (Figure 5-56) where the members of the group can be listed by performing a member search. 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). To list the members of the group, select the type of search to perform from the dropdown: all members, immediate members, effective members, or composite members. Click the **List Members** button. The group members that meet the membership criteria are listed in the table in the lower portion of the screen.



Successfully listed group members!!!

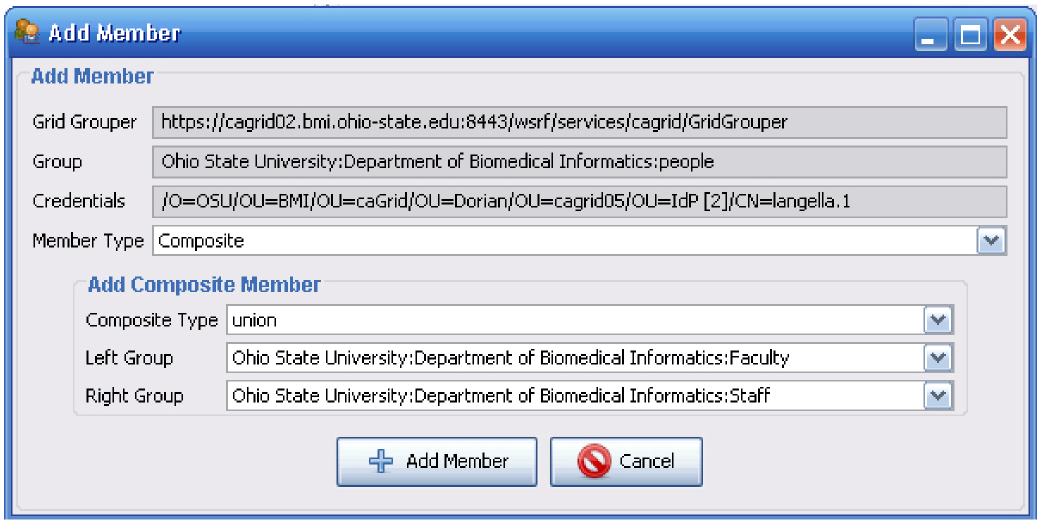
Grid Grouper: https://cagrid02.bmi.ohio-state.edu:8443/wsrf/services/cagrid/GridGrouper
Name: Ohio State University:Department of Biomedical Informatics:Staff
Credentials: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [2]/CN=langella.1

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 5-56 Grid Grouper: Administrating Group Memberships UI

To add additional members to a group click the **Add Member** button. The **Add Member** window opens (Figure 5-57). To add a new member, select the Member Type from the **Member Type** drop down. Members can be added as individuals, groups, or composites. Below the **Member Type** drop down, you are prompted for input based on the member type selected. For individuals, enter the grid identity of the user or service you wish to add as a member. If adding a group as a member, select the group to add as a subgroup. If adding a composite member as shown in Figure 5-57, select a composite type (union, intersection, complement) and two groups that will comprise the composite statement. In the example in Figure 5-57, the user is adding a composite member consisting of the union of the staff group and the faculty group. Once added, this group will contain all the members of both the staff and faculty groups. Note that a group with a composite membership, also referred to as a composite group, may only have one membership (that is, the composite defined). 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 5-57 Grid Grouper: Adding Members to a Group UI

To remove immediate members from a group through the GAARDS UI, use the following steps.

1. Select the member to remove and click the ***Remove Member*** button.
2. Members whose membership to a group is obtained through being a member of a subgroup (Effective Membership) and whose membership is obtained through a composite cannot be directly removed using this method. To remove effective members of a group, the member must be removed from the subgroup of which they are immediate members.
3. To remove composite members from a group, the composite membership associated with the group must be removed. Click 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. To achieve this goal, an interface is defined that should be implemented by an Identity Provider (IdP) service. The framework implementation is provided that exposes the Common Security Module (CSM) as an IdP.

This section describes how to

- Configure the service
- Deploy to a container
- Configure CSM

Configuring the Service

Table 5-4 contains the properties in the file `SRC/deploy.properties` that must be edited.

Property	Description
index.service.url	The URL of the index service to advertise to.
csm.app.context	The name of the application context that contains the CSM

Property	Description
	authentication policy. This value must map to an application name specified in the JAAS configuration file.
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.
saml.provider.pwd	The password for the private key. (If there is no password, this value is ignored.)

Table 5-4 Properties in SRC/deploy.properties

Deploying to the Container

The Authentication Service may be deployed to either the Globus standalone container or the Globus web application (deployed in Tomcat 5.0.28). It does not require Globus to run as 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

A file named `.java.login.config` must be placed in the home directory of the user account that Tomcat or the Globus container are running under. This file must contain entry with an application name that maps to the one specified as the value of `csm.app.context` in the file `SRC/deploy.properties`. Figure 5-58 contains an example JAAS configuration file.

```
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";
};
```

Figure 5-58 Example JAAS configuration file for configuring CSM

The version of CSM used by caGrid requires that a system property named *gov.nih.nci.security.configFile* be set and its value must be the absolute path to a `CSM ApplicationSecurityConfig.xml` file. Figure 5-59 contains an example file.

```
<security-config>
   <upt-context-name>UPT
   </upt-context-name>
   <application-list>
      <application>
         <context-name>myapp
         </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>
            </authorization-provider-class>
            <hibernate-config-file>
            </hibernate-config-file>
         </authorization>
```

```
     </application>
   </application-list>
</security-config>
```

Figure 5-59 `CSM ApplicationSecurityConfig.xml` *file*

To set the system property for the Tomcat container, place the following line in `CATALINA_HOME/catalina.sh.`

```
JAVA_OPTS="$JAVA_OPTS -Dgov.nih.nci.security.configFile= \
/path/to/ApplicationSecurityConfig.xml"
```

To set the system property for the Globus container, place the following line in `GLOBUS_LOCATION/globus-start-container.`

```
updateOptions "gov.nih.nci.security.configFile" \
"/path/to/ApplicationSecurityConfig.xml"
```

Authorization Management

The main responsibility of the Authorization (Authz) component is to provide an integration point between local authorization policy and grid-wide authorization policy. Authorization policy in caGrid is based on membership in groups that are defined in Grid Grouper. Authorization policy within an organization is usually based in an individual's identity within that organization. The Authz component provides a framework to integrate groups that have been defined in the Common Security Module (CSM), which is the authorization policy management system, used by the NCICB and Grid Grouper groups. The result is that local 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 3.1 systems that use CSM 3.1 can plug in the Authz component without changing code.

This section describes how to configure CSM for a caCORE service to use the Authz component.

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 and the authentication manager.

Figure 5-60 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>
   gov.nih.nci.cagrid.authorization.impl.CSMGridAuthorizationManager
            </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 5-60 `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 5-5 contains the jars that must be added to the web applications classpath.

From the Globus 4.0.3 WS Core Distribution	**From other caGrid 1.0 Projects (these end up in cagrid-1-0/ext/lib, when building the Authz project).**	**From the Authz project's lib folder:**
Addressing-1.0.jar	caGrid-1.0-core.jar	c3p0-0.8.5.2.jar
axis.jar	caGrid-1.0-gridca.jar	clm.jar
cog-axis.jar	caGrid-1.0-gridgrouper-client.jar	csmapi.jar
cog-jglobus.jar	caGrid-1.0-gridgrouper-common.jar	hibernate-3.0.5.jar
cryptix-asn1.jar	caGrid-1.0-gridgrouper-stubs.jar	spring-core.jar
cryptix.jar	caGrid-1.0-metadata-common.jar	spring-beans.jar
Cryptix32.jar	caGrid-1.0-metadata-security.jar	
jce-jdk13-125.jar	caGrid-1.0-ServiceSecurityProvider-client.jar	
jgss.jar	caGrid-1.0-ServiceSecurityProvider-common.jar	
puretls.jar	caGrid-1.0-ServiceSecurityProvider-service.jar	
wsrf_common.jar	caGrid-1.0-ServiceSecurityProvider-stubs.jar	
wsrf_core_stubs.jar	cglib-nodep-2.1_3.jar	
wsrf_core.jar	Grouper.jar	
wss4j.jar	Mobius_common_client.jar	

From the Globus 4.0.3 WS Core Distribution	From other caGrid 1.0 Projects (these end up in cagrid-1-0/ext/lib, when building the Authz project).	From the Authz project's lib folder:
	mobius_factories.jar	
	Mobius_gme_client.jar	
	mobius_mako_client.jar	
	Mobius_tools.jar	
	subject-0.2.1.jar	

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

Note: The `clm.jar` and `csmapi.jar` files are from a pre-CSM 3.2 release. Though the official CSM 3.2 versions should work, at the time of writing, this has not been tested.

Finally, a file named `ObjectStateLoggerConfig.xml` must be added to the classpath. That file should look like the following (Figure 5-61).

```
<?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</obj
ect-name>

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

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

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

      <object-
```

```
name>gov.nih.nci.security.authorization.domainobjects.ProtectionElement</obje
ct-name>

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

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

      <object-
name>gov.nih.nci.security.authorization.domainobjects.ProtectionGroupRoleCont
ext</object-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.UserGroupRoleProtectonG
roup</object-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.ProtectionGroupProtecti
onElement</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 5-61 `ObjectStateLoggerConfig.xml`

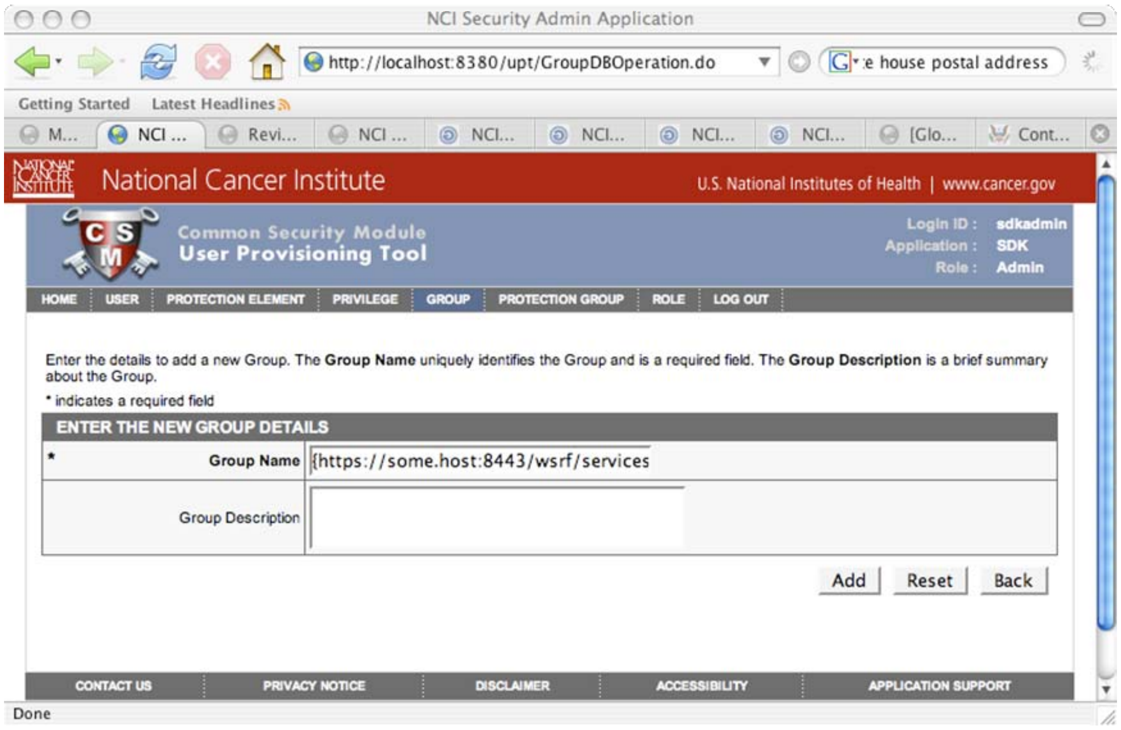
CSM Administration

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 5-62).



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.

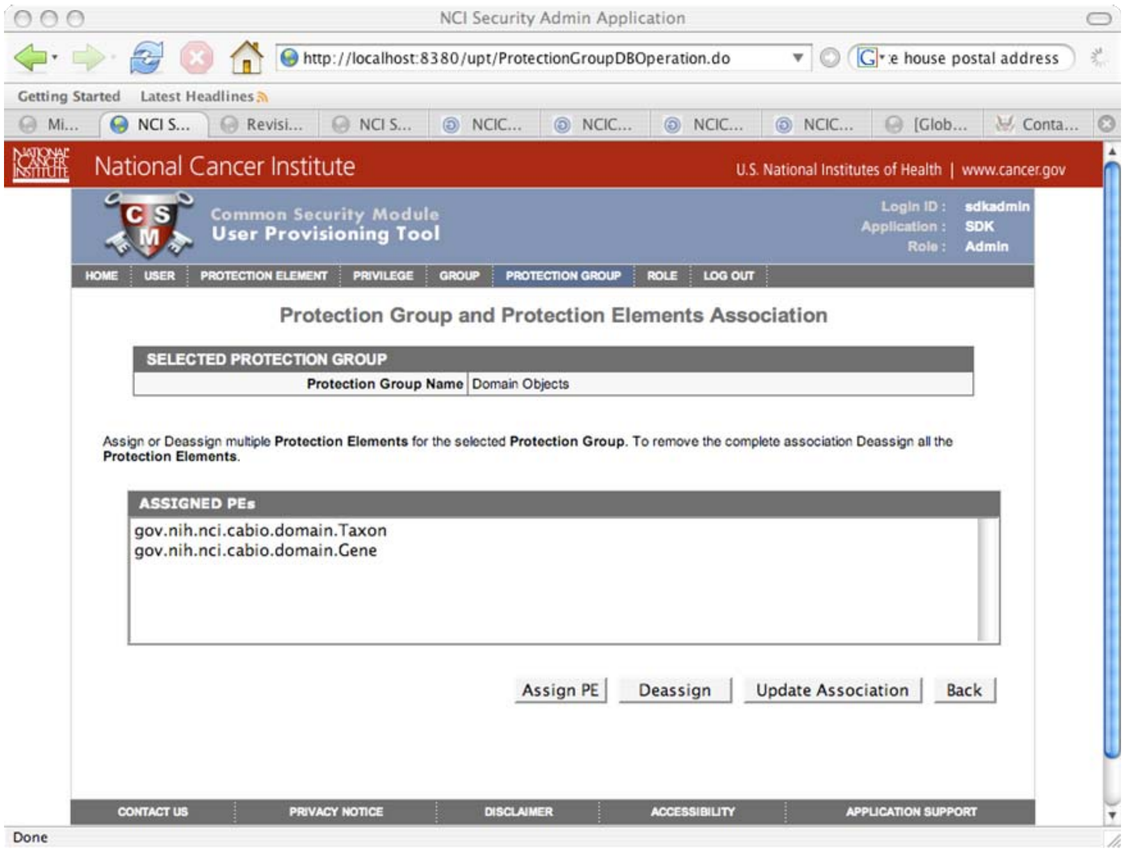
* indicates a required field

ENTER THE NEW GROUP DETAILS

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

Figure 5-62 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 5-63).



Protection Group and Protection Elements Association

SELECTED PROTECTION GROUP	
Protection Group Name	Domain Objects

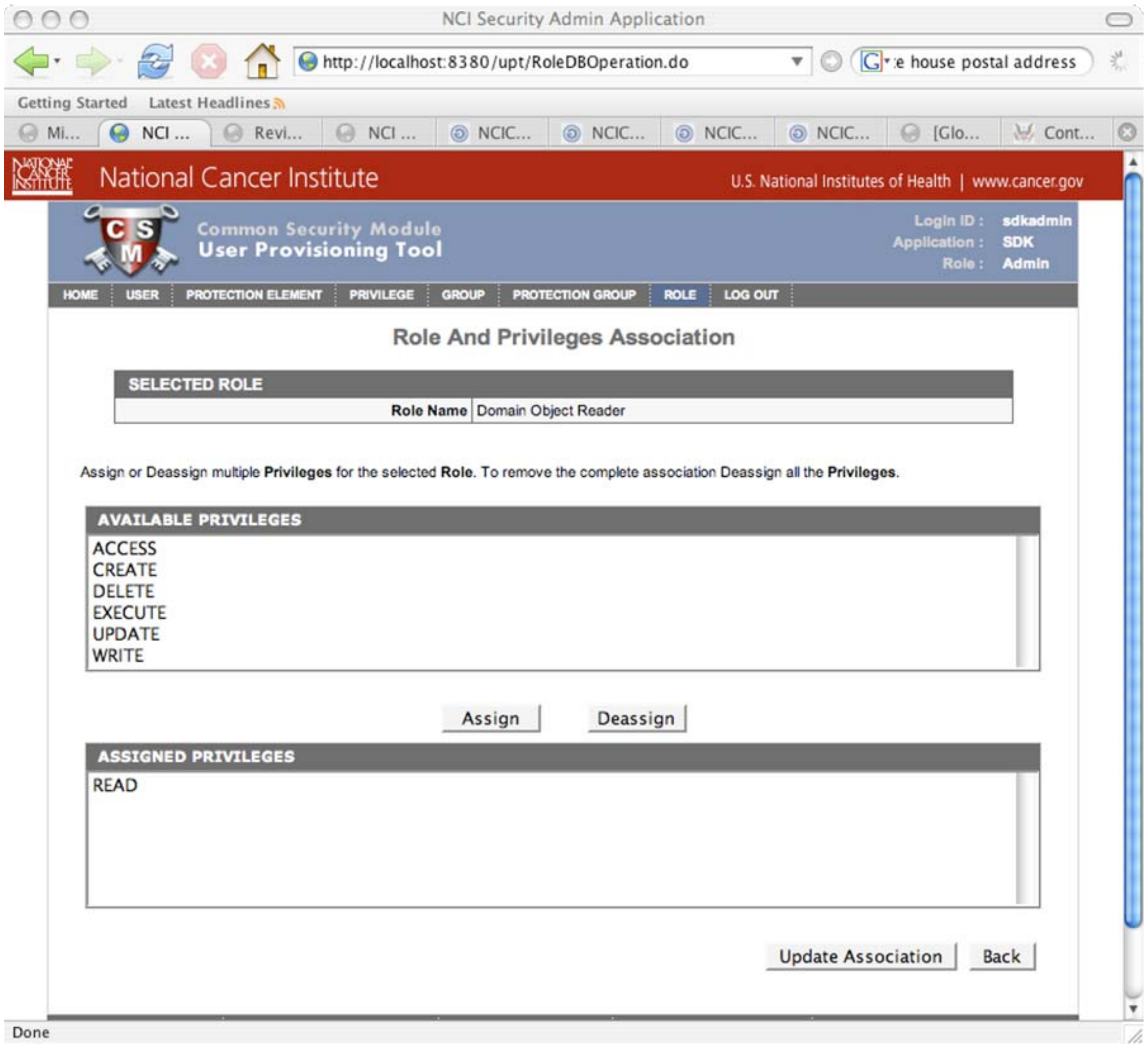
Assign or Deassign multiple **Protection Elements** for the selected **Protection Group**. To remove the complete association Deassign all the **Protection Elements**.

ASSIGNED PEs

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

Figure 5-63 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 5-64).



Login ID : sdkadmin
Application : SDK
Role : Admin

Role And Privileges Association

SELECTED ROLE

Role Name	Domain Object Reader

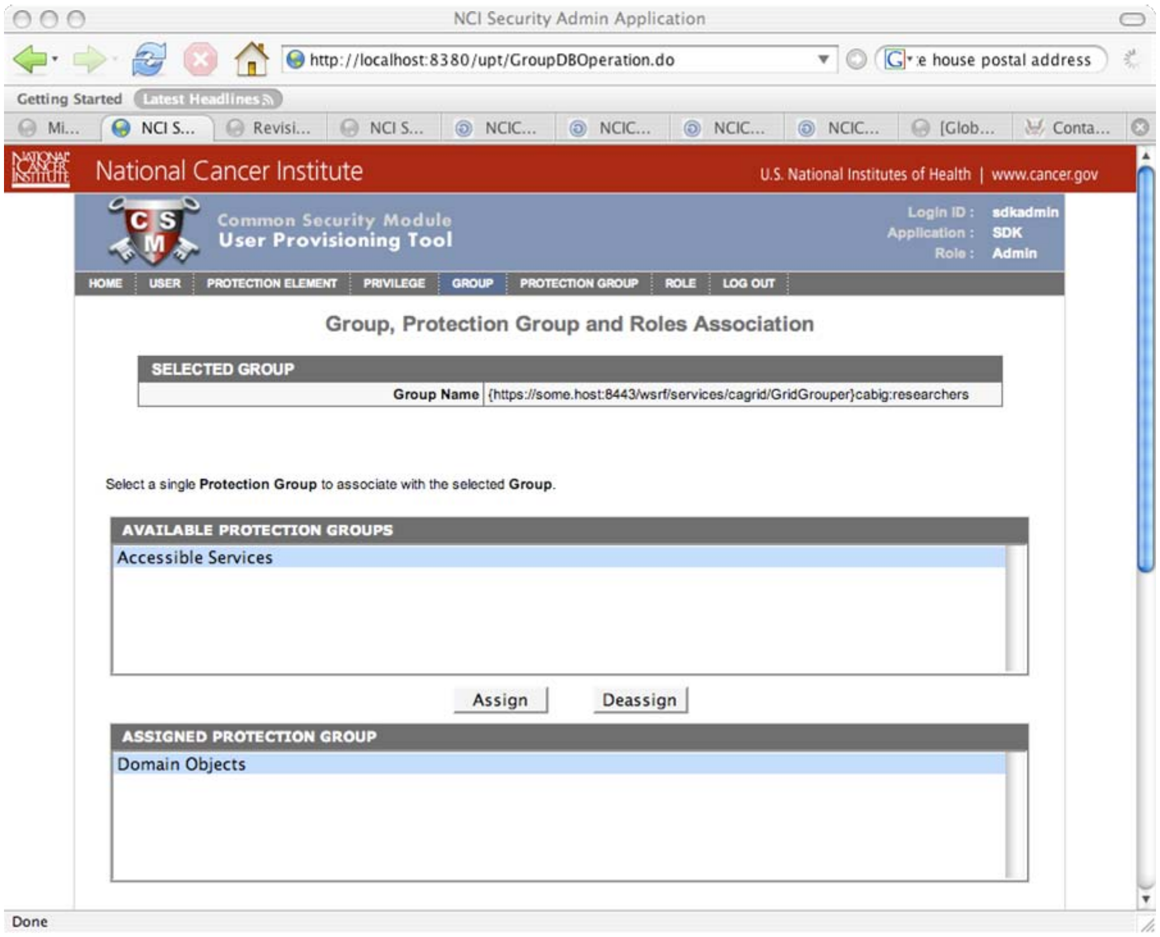
Assign or Deassign multiple **Privileges** for the selected **Role**. To remove the complete association Deassign all the **Privileges**.

AVAILABLE PRIVILEGES
ACCESS
CREATE
DELETE
EXECUTE
UPDATE
WRITE

ASSIGNED PRIVILEGES
READ

Figure 5-64 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 5-65 and Figure 5-66).


NCI Security Admin Application
http://localhost:8380/upt/GroupDBOperation.do
National Cancer Institute
U.S. National Institutes of Health | www.cancer.gov
Common Security Module
User Provisioning Tool
HOME
USER
PROTECTION ELEMENT
PRIVILEGE
GROUP
PROTECTION GROUP
ROLE
LOG OUT
Assign
Deassign
Done

Login ID : sdkadmin
Application : SDK
Role : Admin

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 5-65 Group, Protection Group and Roles Association in the UPT

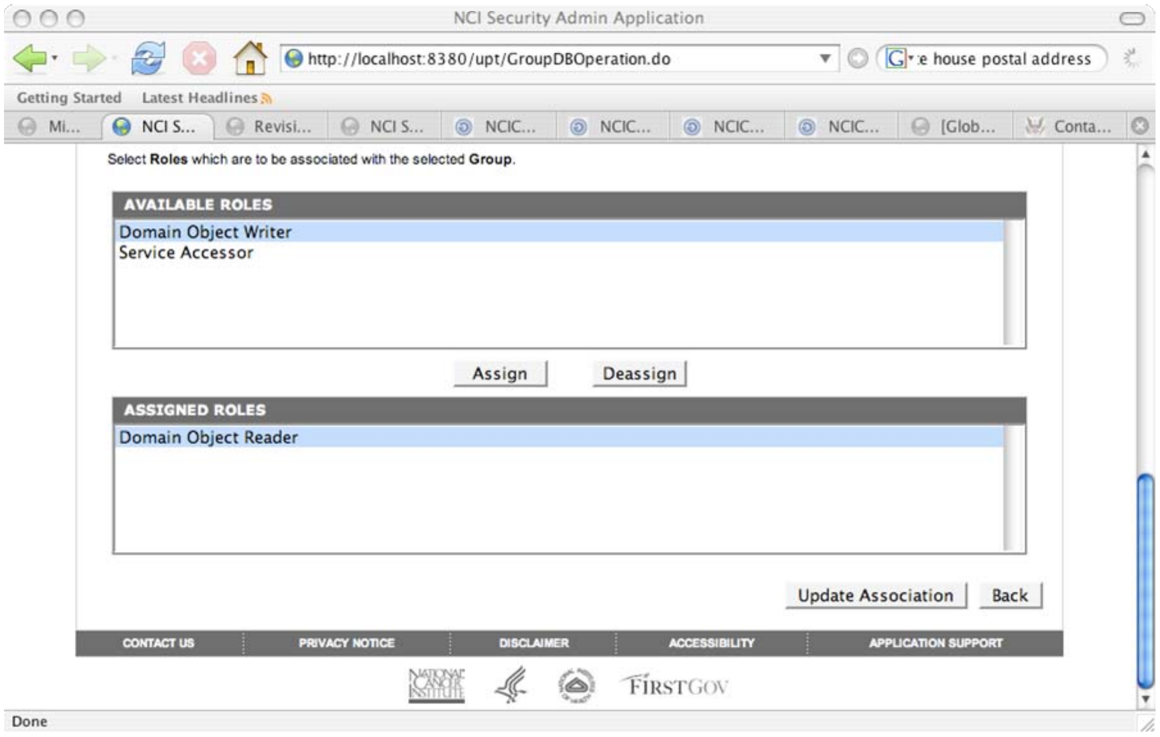


Figure 5-66 Group, Protection Group and Roles Association in the UPT

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 5-67).

```
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 5-67 Code verification

Chapter 6 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:

- Overview on this page
- The Business Process Execution Language (BPEL) on this page
- Creating a Workflow on page 136
- Creating a Simple Workflow Using Example Services on page 136

Overview

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. Here, we define workflow 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 invoke 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" – they contain all of the execution steps that represent a cohesive unit of work. The focus of this document will be on executable processes, as they are concrete workflows that are runnable 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 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 Workflow

Use the following steps to create a workflow.

1. Get the endpoints of the services you want to the workflow to consist of. These endpoints can be obtained from a query to the Index Service based on the ServiceMetaData, though that must be done prior to creating the workflow.
2. Define PartnerLinks for the services you want to interact
3. Create a BPEL document (using a GUI if available)
4. Submit the BPEL document to the WorkflowFactoryService using the command-line client, specifying any input files you need for the workflow
5. The command-line client submits the workflow and starts it.

Creating a Simple Workflow Using Example Services

We will create a workflow that would orchestrate two caGrid services that are built using Introduce. The workflow will take a String as input. Though caBIG The workflow engine invokes the first service that's taking part in the workflow with the input string. The first service appends a string to the input string and returns that as a result. The workflow engine then invokes the second caGrid compliant service with the result of from the first invocation. For simplicity, the second service also appends another string to the input string and itself be exposed as a service

1. Deploy the two test services from the workflow test services distribution
   a. The names of the services are WorkflowTestService1 and WorkflowTestService2
   b. They can be found at https://gforge.nci.nih.gov/frs/download.php/1375/cagrid-1.0-Workflow_Test_Services.tar.gz
   c. Run "ant deploy" to deploy them to Tomcat. Now you have two services that you can orchestrate.
2. Normally you would need to write a little bit of WSDL now to make the services into partnerLinks that can be used in writing a workflow. The workflow test services already have the appropriate partnerLink statements. This step will be automated in the next iteration.

Submitting the Workflow

You can use the command-line client bundled with the release to submit the workflows. The workflow client takes arguments for the workflow factory location (EPR) and the location of the BPEL file. The client then submits the workflow to the factory service, creates the workflow resource, and returns the EPR for workflow management service the to the user. At this point the workflow definition is validated and found to be syntactically and semantically right. Otherwise exceptions are thrown to the user specifying the nature of error. The workflow management service is used for starting, stopping, pausing, and canceling the workflow. Another Command-line client is provided to execute the submitted workflows.

Executing the Workflow

The EPR of the workflow resource that is returned from the above step can be used to execute the workflow process created using input data (if the workflow needs). The command-line client takes the EPR that the user provides, locates the Workflow Service and invokes the start operation on it with the input parameters. This is a non-blocking call and returns immediately. The Workflow Service provides other operations for the user, such as querying for the status of the workflow.

Query the Status of the Workflow

The client used in step above can be used to query the status of the workflow if the user provides the EPR of the workflow resource that he wants to find status for. This operation returns one of the five states a workflow can be in. It throws an exception if the workflow cannot be found.

Terminating a Workflow

Since the workflows are modeled as WS-RF 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 a Workflow

The command-line client provides an operation by which users can pause an active workflow. This command will result in the service invoking the “pause” operation of the workflow management service using the workflow id.

Resuming a Paused Workflow

The command-line client provides an operation by which users can resume a paused workflow. This command will result 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 3.1 Technical Guide", ftp://ftp1.nci.nih.gov/pub/cacore/caCORE3.1_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*
`{jboss-home}`	The base directory where JBoss is installed on the server
API	Application Programming Interface
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
caMOD	Cancer Models Database
cardinality	Cardinality describes the minimum and maximum number of associated objects within a set
CDE	Common Data Element
CGAP	Cancer Genome Anatomy Project
CMAP	Cancer Molecular Analysis Project
CN	Common Name
CS	Classification Scheme
CSI	Classification Scheme Item
CSM	Common Security Module
CTEP	Cancer Therapy Evaluation Program
CUI	Concept Unique Identifier
CVS	Concurrent Versions System
DAIS	Data Access and Integration Services
DAML	DARPA Agent Markup Language
DAO	Data Access Objects
DARPA	Defense Advanced Research Projects Agency
DAS	Distributed Annotation System
DL	Description Logic
EA	Enterprise Architect
EBI	European Bioinformatics Institute
EVS	Enterprise Vocabulary Services
GAI	CGAP Genetic Annotation Initiative

Term	***Definition***
GEDP	Gene Expression Data Portal
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.
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.
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
JET	Java Emitter Templates
JMI	Java Metadata Interface
JSP	JavaServer Pages
JUnit	A simple framework to write repeatable tests (http://junit.sourceforge.net/)
LDAP	Lightweight Directory Access Protocol
LLT	Lowest Level Term
LOINC	Logical Observation Identifier Names and Codes
MAGE	MicroArray and Gene Expression
MAGE-OM	MicroArray Gene Expression - Object Model
MDA	Model Driven Architecture
MedDRA	Medical Dictionary for Regulatory Activities
metadata	Definitional data that provides information about or documentation of other data.
MGED	Microarray Gene Expression Data

Term	***Definition***
Mobius	An array of tools and middleware components to coherently share and manage data and metadata in a Grid and/or distributed computing environment.
multiplicity	Multiplicity of an association end indicates the number of objects of the class on that end may be associated with a single object of the class on the other end
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
OIL	Ontology Inference Layer
OilEd	Ontology editor allowing you to build ontologies using DAML+OIL
OLLT	Obsolete Lower Level Terms
OMG	Object Management Group
ORM	Object Relational Mapping
PT	Preferred Term
RDBMS	Relational Database Management System
SDE	Service Data Element
SDK	Software Development Kit
Semantic connector	A development kit to link model elements to NCICB EVS concepts.
SOA	Service Oriented Architecture: A discipline for building reliable distributed systems that deliver application functionality as services with the additional emphasis on loose coupling between interacting services.
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	System Organ Class
SPORE	Specialized Programs of Research
SQL	Structured Query Language

Term	***Definition***
SSC	Special Search Categories
UI	User Interface
UID	User Identification
UML	Unified Modeling Language
UML	Unified Modeling Language
UMLS	Unified Medical Language System
UPT	User Provisioning Tool
URL	Uniform Resource Locators
VD	Value Domain
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
WSDL	Web Services Description Language
WSRF	Web Services Resource Framework
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
XML	Extensible Markup Language
XPath	XML query/traversal language adhering to the XPath specification set forth by the W3C.
XQuery	XML query/transformation language adhering to the XQuery specification set forth by the W3C.

Index

Analytical and data services
  overview, 7
Analytical service developer, 4
Apache Axis
  relationship to Introduce, 8
ApplicationSecurityConfig.xml, 125
Authentication management. *See CSM*
Authorization Management. *See* Authz
Authz
  configuring JAAS, 125
  overview, 125
  required jars, 127
  setting classpath, 127
caBIG
  references, 143
caCORE
  references, 143
caCORE SDK, 41
caGrid
  client APIs, 50
  overview, 49
  user management, 86
caGrid document, 3
caTRIP
  client application case study, 55
  definition, 55
Client API
  discovery API, 52
  EPR, 51
  inspecting metadata, 53
  Introduce, 19
  invoking operations on service, 54
  obtain EPR for service, 51
  overview, 50
  secure communication, 50
Client application developer, 4
Create basic service, 7
CSM
  administration example, 130
  configuring, 123
  configuring a service, 122
  deploying to container, 123
  overview, 122
Data service developer, 4
Deploy
  grid service, 19
Deploy service, 7
Deploying
  Dorian, 95
  Grid Grouper, 111
Discover service methods, 10
Discovery API, 52
DiscoveryClient, 53
Document conventions, technical guide, 2
Dorian
  adding trusted identity provider, 99
  building, 90
  certificate authority configuration, 93
  configuring, 90
  definition, 60
  deploying, 95
  identity management configuration, 92
  identity provider configuration, 91
  managing grid proxies, 96
  managing grid user accounts, 101
  managing grid users, 89
  managing IdP users, 106
  managing trusted identity providers, 97
  registering with IdP, 105
  removing trusted identity provider, 101
  setting default administrator account, 95
  software prerequisites, 89
  viewing/updating trusted identity provider, 100
End point reference
  definition, 51
  obtaining for service, 51
Enterprise Vocabulary Services, 49
Examples
  caTRIP invocation adn client API, 56
  caTRIP metadata and client API, 56
  client API and caTRIP discovery, 55
  create basic analytical service, 20
  create grid service using data extension, 34
  create service using caCORE SDK, 41
  CSM administration, 130
  discovering services, 53
  inspecting metadata, 54
GAARDS
  definition, 60
Global Model Exchange
  relationship to Introduce, 8
Globus toolkit
  removing administrator, 71
Globus Toolkit, 64
  adding administrator, 70
  adding authority, 81
  adding trust levels, 73
  adding trusted certificate authorities, 75
  building, 67
  building SyncGTS, 86
  configuring, 67
  deploying, 68

managing certificate authorities, 74
managing trust levels, 71
prioritizing authority, 83
relationship to Introduce, 8
removing authority, 83
removing trust levels, 74
removing trusted certificate authorities, 79
running SyncGTS, 86
software prerequisites, 67
synchronization, 84
syncing with trust fabric, 83
updating an authority, 82
user management, 86
view/modify trust levels, 72
viewing/modifying trusted certificate authorities, 77
Grid Grouper
administering, 111
adminstrating groups, 117
building, 110
configuring, 110
deploying, 111
overview, 108
software prerequisites, 109
stem administration, 112
Grid Security Infrastructure, 64
Grid trust fabric
definition, 62
Grid Trust Fabric
bootstrapping, 66
managing, 69, 79
Grid Trust Service, 64
Groups
adminstrating in Grid Grouper, 117
memebership, 120
privileges, 118
Hibernate
configuring for Authz, 126
hibernate.cfg.xml, 126
Introduce
client API, 19
discover service methods, 10
graphical development environment, 9
overview, 8
service creation, 10
service modification, 12
ISO/IEC 11179, 49
Metadata
inspecting, 53
Metadata API, 53
Modify service, 7
References
caBIG, 143
caBIG materials, 143
caCORE, 143
caCORE material, 143
technical manuals, guides, 142
Referneces
scientific publications, 139
Service
contexts, 17
create basic analytical, 20
create using caCORE SDK, 41
create using data extension, 34
deployment, 19
levels of security, 16
operations, 14
resource properties, 15
service properties, 15
types, 13
Service administration
administration user interface, 61
bootstrapping grid trust fabric, 66
Globus toolkit, 64
grid trust fabric, 62
grid trust service, 64
overview, 59
security infrastructure, 60
Service administrator, 5
Service creation, 10
Service developer, 3
Service modification, 12
Software prerequisites
Dorian, 89
Globus, 67
Grid Grouper, 109
Stems
administrating in Grid Grouper, 112
managing child groups, 116
managing children, 115
privileges, 113
SyncGTS
building, 86
running, 86
Text conventions, technical guide, 2
User accounts
managing, 103
removing, 104
User role
analytical service developer, 4
client application developer, 4
data service developer, 4
definition, 3
service administrator, 5
service developer, 3