



## caGrid Technical Overview

### caGrid Technical Overview for Software Developers Using caGrid

#### The caGrid Knowledge Center

#### Document Revision History

Revision	Date	Author(s)	Change Reference	Reason for Change
1.0	08/10/2008	Justin Permar and Tahsin Kurc		Initial Document
1.0.1	08/11/2008	Tahsin Kurc		Modified formatting and added references.



Contents

- 1. Introduction..... 3
- 2. caGrid Framework ..... 5
  - 2.1. Interoperable and Model Driven ..... 6
  - 2.2. Semantically Discoverable ..... 8
  - 2.3. Secure and Manageable ..... 9
- 3. caGrid Architecture Components ..... 12
  - 3.1. Introduce Toolkit..... 12
  - 3.2. caGrid Query Language ..... 15
  - 3.3. Types of Grid Services ..... 15
    - 3.3.1. Data Services..... 15
    - 3.3.2. Analytical Services..... 16
  - 3.4. Security..... 17
    - 3.4.1. Grid Account Management and Federation ..... 17
    - 3.4.2. Authorization: Grid Service Access Control ..... 19
    - 3.4.3. Grid Trust Management ..... 20
    - 3.4.4. Credential Delegation Service (CDS) ..... 20
  - 3.5. Large Data Transfer ..... 21
  - 3.6. Web Integration..... 21
    - 3.6.1. webSSO..... 21
    - 3.6.2. caGrid Portal ..... 22
  - 3.7. caGrid Coordination Services..... 22
    - 3.7.1. Index Service and Metadata ..... 22
    - 3.7.2. Federated Query and Workflow ..... 23
    - 3.7.3. Global Model Exchange (GME) Service ..... 24
    - 3.7.4. caGrid Services for caDSR and EVS..... 24

### 1. Introduction

This document presents a technical overview of the caGrid middleware, its core components, and how the caGrid components can be used to build applications. The caGrid infrastructure[1-3] is designed to facilitate interoperability and federation of information and analytical resources, potentially developed by independent groups, in a multi-institutional environment. caGrid provides tools and APIs for software developers to build secure, interoperable services and applications. Sharing data and analytical routines with collaborators provides researchers with the capability to benefit from the combined expertise, knowledge, and resources of multiple organizations.

caGrid combines Service Oriented Architecture (SOA), Grid computing, and the Model Driven Architecture (MDA) (<http://www.omg.org/mda/>) in an integrated framework. Service Oriented Architecture is a distributed computing architecture where functionality of a component is encapsulated as a service and deployed on the network multiple services can be used together to produce additional and more complex functionality than that of a single service. Most SOA systems employ Web Services technologies as the underlying platform. Web Services provides access to services via standard web protocols. Client applications access services via standardized service APIs.

Grid computing refers to the concept of utilizing distributed resources[4-9]. Grid computing started as a mechanism to enable access to high-end computing facilities across multiple supercomputer centers to solve complex scientific and engineering problems that require massive computing and storage resources. Grid computing has since evolved into a platform that facilitates the development, deployment, and secure federation of data and analytical resources. Following the SOA standardization effort, the Grid computing community developed the Open Grid Services Architecture (OGSA) standards, which adapt and extend Web Services standards for scientific applications. The OGSA standards define mechanisms and guidelines for such additional features as stateful services, service notification, and management of service lifetime, within the context of established Web Services standards. The OGSA has evolved into the Web Services Resource Framework (WSRF)[10, 11]. The WSRF is based on the core concepts underlying the OGSA; however, it extends these concepts and adds new functionality to establish a path towards unifying Web services and Grid services technologies. We refer the reader to the following references for additional background information on Grid computing and SOA.

- Service Oriented Architecture: [http://en.wikipedia.org/wiki/Service-oriented\\_architecture](http://en.wikipedia.org/wiki/Service-oriented_architecture)
- Grid Computing: <http://www.globus.org/alliance/publications/papers/anatomy.pdf>
- Web Services Resource Framework (WSRF): <http://www.globus.org/wsrf>

caGrid builds on the WSRF standards. Software developers using caGrid build “Grid services”, i.e., each analytical and data resource is wrapped as a Grid service and can be invoked via WSRF protocols. A distinguishing feature of caGrid over the basic SOA and Grid computing approaches is the improved support for *syntactic* and *semantic* interoperability among Grid services. *Syntactic interoperability* enables a consumer (e.g., a client program) to programmatically access a resource (e.g., a service). The

major obstacles to syntactic interoperability are the heterogeneity of the programming and messaging interface syntax and data structures that encapsulate the same type of data across different systems. That is, a software system cannot access the components of another system unless 1) there are programmatic interfaces to those components, 2) these interfaces and how they can be invoked are well-defined, and 3) the two systems have agreed on the structure of data types exchanged between the components of the two systems. *Semantic interoperability*, on the other hand, is concerned with use of a resource – i.e., semantically correct interpretation and consumption of a resource. In a distributed environment, in which resources are developed by independent groups, the meaning of a resource and the attributes of data structures representing the content of the resource can be named and described differently by different groups. Two data elements representing the same entity might have been defined using different terms; more importantly, two data elements representing different concepts may have the same attributes and attribute names. The contents and meaning of a resource need to be explicitly defined using terms from a vocabulary, which is agreed upon by the community (resource providers and resource consumers) in order to address obstacles to semantic interoperability.

caGrid adopts a Model Driven Architecture approach to enable interoperability through object-oriented abstractions, common data elements, and controlled vocabularies. That is, client and service APIs in caGrid are object-oriented and operate on well-defined objects. These objects, in turn, are built from common data elements and controlled vocabularies registered on the Grid. A caGrid-compliant data service abstracts a data source's data elements as objects. Similarly, an analytical resource (e.g., an analysis program) implemented as a caGrid analytical service provides methods that input objects and return objects.

In addition to resource-specific data and analytical services, caGrid provides Grid-wide coordination services that are used by both Grid client and other Grid services. Coordination services include metadata management services, advertisement and discovery services, federated query services, workflow execution services, and security services. For instance, the advertisement and discovery services can be used to both advertise and discover the availability, service operations, and state of services deployed to the Grid. Coordination services can be replicated and distributed in the environment to achieve higher availability and performance. Figure 1 shows the caGrid infrastructure with coordination services (e.g., metadata services, security services, workflow service) as well as community-provided services. Users can access these services through web portals or client applications.

## caGrid Technical Overview

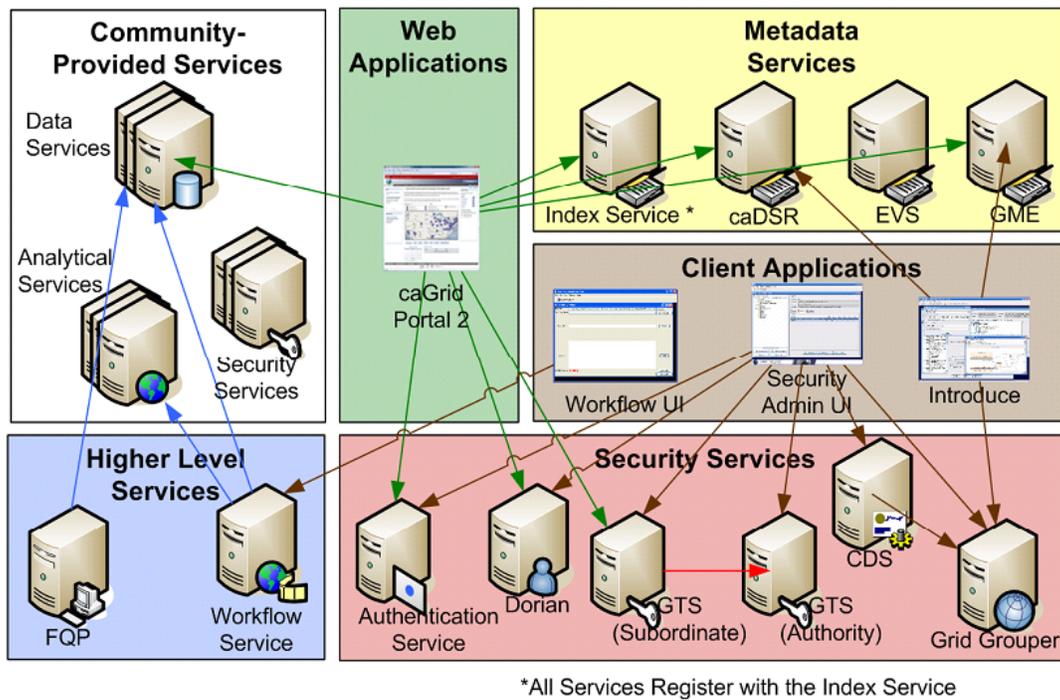


Figure 1. caGrid infrastructure and environment.

## 2. caGrid Framework

The design of caGrid is mainly driven by the requirements and use cases from the biomedical research community. While numerous complex use cases are possible, common requirements across different use cases include the need to: 1) carry out searches that return precisely defined attributes and data values from heterogeneous information sources, 2) access, aggregate, and perform integrated analysis of multiple types of data, and 3) enforce access control policies surrounding shared Grid resources. In order to support these requirements, caGrid facilitates programmatic discovery of the structure and semantics of Grid resources. Semantic discovery is a powerful mechanism for researchers to find Grid resources relevant to their work.

caGrid leverages Grid Services technologies, including the Globus Toolkit (<http://www.globus.org>) and Mobius (<http://projectmobius.osu.edu/>)[12], and tools developed by the NCI, such as the caCORE infrastructure (<http://ncicb.nci.nih.gov/infrastructure/cacoresdk>)[13, 14], to deliver functionality. While the caGrid 1.x infrastructure is built upon Globus Toolkit 4.0 (GT4)[5], it aims to be programming language and toolkit agnostic. Specifically, caGrid services are standard WSRF v1.2 services and can be accessed by any specification-compliant client. The architecture of caGrid has several important features to better address the informatics needs of biomedical research. In the rest of this section we present an overview of these features.

### 2.1. Interoperable and Model Driven

Projects which involve information integration and analysis using heterogeneous data and analytical resources developed by different groups have to tackle several challenging informatics problems. The first problem is that of programmatically accessing resources that may be hosted at other institutions. Multiple institutions have different security infrastructures; data to be accessed may be stored in various database management systems, requiring multiple query and retrieval mechanisms; analytical programs may have to be invoked in different ways. The second problem is resources may have different data representations. Each resource may even define the same conceptual data type using a different data structure and database schema than other resources. A software developer would need to find out how a data type is represented in each resource and would have to implement software to consume differing data structures – in such an approach, addition of a new source might require changes to the client application and other resources that consume or produce the data type. The third problem is that each resource provider may use a different terminology and ontology to annotate their data and data types. It is possible that data types encoding the same entity may have been annotated with terms from different ontologies. Multiple information providers may have different definitions of a concept and/or choose different vocabulary to define the same concept. These differences make it difficult to interpret information received from a resource and carry out computational reasoning and semantic integration on the data.

Web services standards solve the programming language interoperability problem (C# and Java, for example) by specifying language-independent access to distributed resources. WSRF solves additional interoperability issues by defining standardized web service interfaces. In addition to these standards, controlled vocabularies, common data elements (CDEs), published information models, and well-defined application programming interfaces (APIs) are necessary to enable syntactic and semantic interoperability among resources. As presented in the introduction section, *Syntactic interoperability* facilitates programmatic access to otherwise heterogeneous resources. *Semantic interoperability* is needed to ensure correct interpretation of a resource and its content. The caBIG™ program and caGrid adopt a model driven architecture approach to enable syntactic and semantic interoperability between resources. The caBIG™ community has developed guidelines and a set of requirements to represent an application's level of interoperability. These guidelines are outlined in the caBIG compatibility guidelines document ([https://cabig.nci.nih.gov/guidelines\\_documentation/](https://cabig.nci.nih.gov/guidelines_documentation/)). These guidelines indicate that a resource should have well defined APIs that provide object-oriented access to backend systems, employ community curated/harmonized terminologies and common data elements for its data models, and expose published domain models. Such resources are considered syntactically and semantically interoperable. When these resources are deployed in a multi-institutional environment, they should be wrapped as Grid services with WSRF compliant service interfaces, use XML for data exchange, and be supported by a common framework for functions such as service advertisement, discovery, security, and invocation. As is described in the introduction section, caGrid services represent an object-oriented view of backend resources. The structure of data served by a data source or consumed/produced by an

analytical resource is represented with a domain model, which is expressed as object classes and associations between them. Client and service APIs in caGrid operate on instances of objects, whose classes and class attributes are registered and published on the Grid.

caGrid leverages existing NCI data modeling infrastructure to manage, curate, and employ domain models. Specifically, a Grid developer creates UML class diagrams to model data that will be shared on the Grid. Using UML tools and NCI data modeling infrastructure, the domain models are converted into common data elements in the form of ISO/IEC 11179 administered components and registered in the Cancer Data Standards Repository (caDSR)[13, 14]. These data elements are annotated by terms and concepts drawn from vocabulary registered in the Enterprise Vocabulary Services (EVS)[13, 14]. The concepts of data elements and the relationships among the data elements thus are semantically described. More information about the modeling process can be found at the caBIG™ Boot Camp website ([https://cabig.nci.nih.gov/training/2007\\_boot\\_camp](https://cabig.nci.nih.gov/training/2007_boot_camp)). In the Grid environment, clients and services communicate using messages encoded in XML. When an object is transferred between clients and services, it is serialized into a XML document that adheres to a registered XML schema. The requirement for use of registered data models and XML schemas is to ensure syntactic and semantic interoperability between two end-points exchanging information. With a published model and schema, the receiving end-point can parse the data structure and interpret the information correctly. XML schemas corresponding to common data elements and object classes are registered in the GME service. In summary, the caDSR and EVS define the properties and semantics of caBIG™ data types, and the GME defines the syntax of their XML materialization.

It should be noted that caGrid provides a flexible development and runtime infrastructure so that a service provider and developer can achieve interoperability in steps. While a service deployed for production use in the caBIG™ environment *must* be syntactically and semantically interoperable – exposing well-defined, registered domain models and objects annotated with terms from controlled vocabularies –, a developer does not have to satisfy all of the requirements at once to be able to develop and test a service. The developer can stand up a local instance of caGrid and develop and deploy services in this instance without having to register data elements in the caDSR, or use the EVS, or register schemas in the GME. A collaborative group can set up a caGrid instance for their project and implement *strongly-typed* services that will be only syntactically interoperable. That is, these services use objects, the corresponding XML schemas of which are registered in the GME, but do not register the definitions of these objects in the caDSR and the EVS. This modularized, flexible infrastructure of caGrid offers an efficient development environment and an efficient mechanism for adoption of the caGrid technologies. It allows communities and groups outside the caBIG™ program, which may not have well established vocabularies and/or resources to support curation and harmonization processes for semantic interoperability, to use caGrid in their projects.

## 2.2.Semantically Discoverable

Grid services register service metadata with a central indexing registry service (Index Service). This service is both a “yellow pages” and “white pages” for the Grid. A researcher can then discover services of interest by looking them up in this registry. The common service metadata, to which every service is required to adhere, contains information about the service-providing cancer center, such as the point of contact for the service and the institution’s name which is providing the service. Data Services provide an additional “domain model” metadata, which details the domain model, including associations and inheritance information, from which the objects being exposed by the service are drawn. These metadata standards leverage the data models registered in caDSR and link them to the underlying semantic concepts registered in EVS. The definitions of the objects in a domain model themselves are described in terms of their underlying concepts, attributes, attribute value domains, and associations to other objects being exposed. The common service metadata also details the objects, used as input and output of the services operations, using the same format as the data service metadata. In this way, all services fully define the domain objects they expose by referencing the data model registered in caDSR, and identify their underlying semantic concepts by referencing the information in EVS. Metadata associated with a caGrid data service can specify, for example, what types of image data are served by the source and whether longitudinal follow-up imagery is available for patients. Metadata for a caGrid analytical service may include information about what analytical methods are exposed by the service, what data types each method takes as input, and what data types each returns as output. caGrid provides a series of high-level APIs and user applications for performing lookup on service metadata which greatly facilitate the discovery process. The advertisement and discovery process is illustrated in Figure 2.

## caGrid Technical Overview

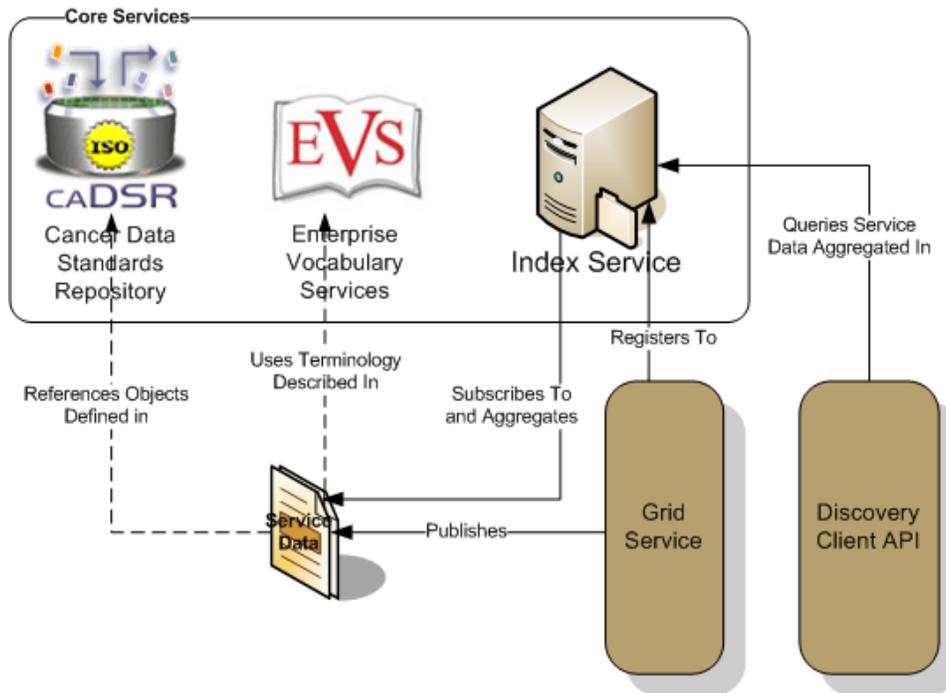
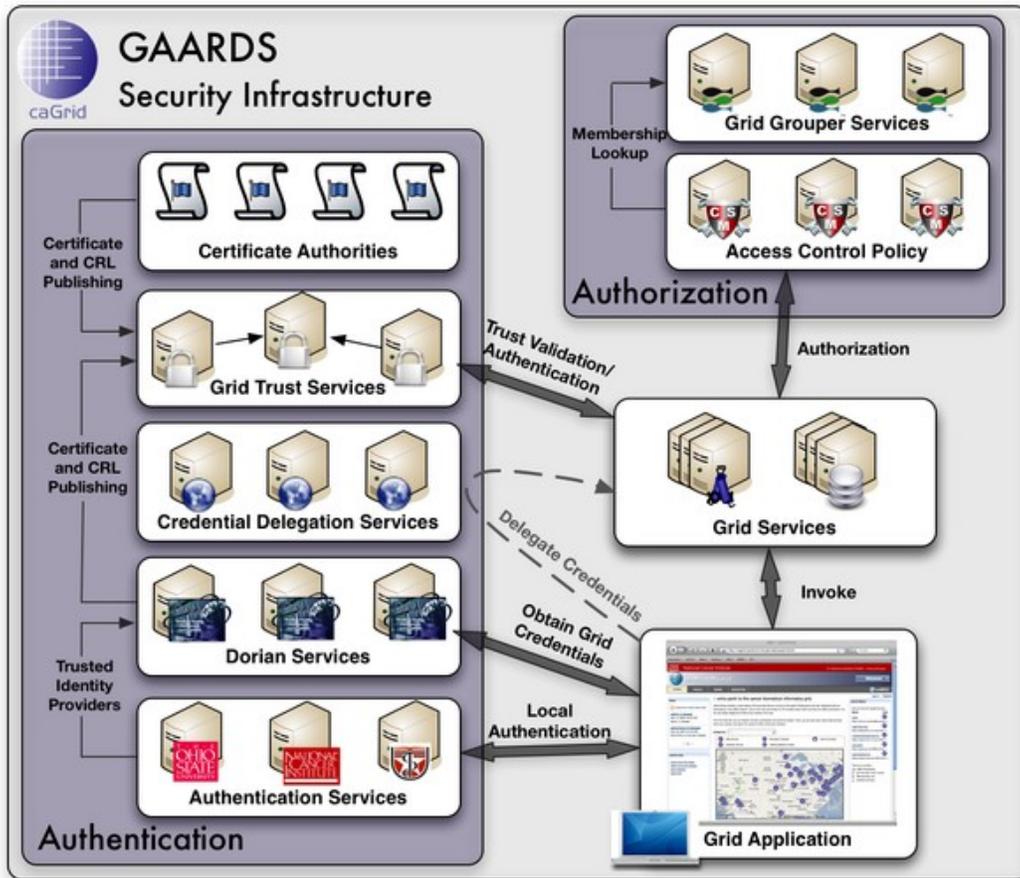


Figure 2. caGrid Discovery Overview

### 2.3. Secure and Manageable

Security is a critical component of caGrid. On the Grid, security requirements exist both for protecting intellectual property and ensuring protection and privacy of patient data and other sensitive information. caGrid components that together provide comprehensive security support are known as the Grid Authentication and Authorization with Reliably Distributed Services (GAARDS) infrastructure [15-18]. 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 Globus Grid Security Infrastructure (GSI) (<http://www.globus.org/security/overview.html>) to provide enterprise services and administrative tools for:

1. Grid user management
2. Identity federation
3. Trust management
4. Group/Virtual Organization management
5. Access control policy management and enforcement
6. Integration between existing security domains and the grid security domain



**Figure 3.** GAARDS Security Infrastructure

The main components of GAARDS are: **Dorian** service for the provisioning and management of Grid users accounts. **Grid Trust Service (GTS)** for maintaining and provisioning a federated trust fabric consisting of trusted certificate authorities, allowing Grid services to make authentication decisions against the most recent information. **Grid Grouper** for a group-based authorization solution for the Grid. **Authentication Service** for issuing SAML assertions for existing credential providers so they may easily integrate with Dorian and other Grid credential providers. **Credential Delegation Service** for a client (the delegator) to be able to express a delegation policy, entitling a prescribed collection of other grid entities (the delegates) to assume the delegator’s identity for a limited time.

In order for users and applications to communicate with secure services, they need Grid credentials. Obtaining Grid credentials requires having a Grid User Account. GAARDS provides an account management and identity provider service, Dorian, and two mechanisms for registering for a Grid user account: 1) registering directly with Dorian or 2) having an existing user account in another trusted security domain (e.g., a participating institution’s security domain). In order to use an existing user account to obtain Grid credentials, the existing credential provider must be registered with GAARDS as a

Trusted Identity Provider. Figure 3 illustrates an example process for obtaining Grid credentials. In this example, the user first authenticates with his/her institution's credential provider and obtains a SAML assertion as proof he is authenticated. The user then uses the SAML assertion provided to obtain Grid credentials from GAARDS. Assuming the institution's credential provider is registered with GAARDS as a trusted identity provider and that the user's account is in good standing, GAARDS Dorian service will issue Grid credentials to the user. (Note: if a user has an account with Dorian, he can contact Dorian directly to obtain Grid credentials). After a user has obtained Grid credentials, he may invoke secure Grid operations.

Upon receiving Grid credentials from a user, a secure service authenticates the user to ensure the credentials are valid. Part of Grid authentication is verifying that Grid credentials are issued by a trusted Grid credential provider (e.g., Dorian). The Grid Trust Service (GTS) of GAARDS maintains the official list of trusted credential providers. This list is known as the "trust fabric". Credential providers are registered as trusted Certificate Authorities (CAs). Trusted CAs periodically publish updated information to the GTS. Grid services authenticate Grid credentials against the trusted CAs (shown in Figure 3).

Once the user has successfully authenticated, a secure Grid service can perform an authorization check to determine if a user is authorized to invoke requested service operations. It is important to note that all authorization decisions are made by the service itself, but GAARDS implements services and tools to support common authorization mechanisms. The GAARDS infrastructure provides two authorization options, which can each be used independently or together to implement authorization policies for a service<sup>1</sup>. The first authorization option is the Grid Grouper service. Grid Grouper provides a group-based authorization solution for the Grid, whereby Grid services and applications enforce authorization policy based on a group membership check. The caCORE Common Security Module (CSM) supports centralized authorization checks. These checks are "centralized" because CSM is deployed specifically for a service that performs the authorization check. CSM policies are constructed by specifying read/write access to protected elements; Grid services using CSM defer authorization checks to CSM. Based on the access control policy maintained in CSM, CSM decides whether or not a user is authorized. In addition, Grid Grouper and CSM can be used together; for example, access control policies specified in CSM can be based on membership to groups in Grid Grouper.

In order to support Grid workflows (a workflow is a group of coordinated services that together provide a desired analysis or other end result), users need the ability to allow another user or service to perform work on their behalf. The Credential Delegation Service (CDS) allows a user (the delegator) to express a delegation policy, entitling a prescribed collection of other grid entities (the delegates) to assume the delegator's identity for a limited time. With GAARDS, a user can log in to Dorian and then invoke Grid workflows by delegating his credential to the CDS; services involved in the Grid workflow retrieve the user's credential to perform work for the user.

---

<sup>1</sup> Other authorization mechanisms also can be employed in conjunction with GAARDS.

### 3. caGrid Architecture Components

The remainder of this document describes various components of caGrid, including tools, APIs, and services provided in the caGrid software release.

#### 3.1. Introduce Toolkit

The caGrid infrastructure employs the Globus Toolkit (GT) as the underlying Grid middleware backbone and runtime environment. The GT provides a suite of core runtime services and tools for developing and deploying Grid services. However, these are low level tools, requiring service developers to understand the details of the GT and how and in what order the tools should be invoked, and to keep track of several files and directories that are required for successful compilation and deployment of services. The Introduce toolkit[19] helps a service developer by coordinating the various steps of the service development and deployment via the tools provided by the GT and by managing the necessary directories and files. It abstracts away the details of invoking the various tools so that the developer is freed up to concentrate on the details of implementing his/her domain-specific code. The main features of Introduce can be summarized as follows:

- It provides a graphical development environment (GDE) and high-level functions that encapsulate and hide the common complexities and low level command-line tools of the GT for generating a suitable service layout. These include client and server wrappers to encapsulate the “boxed” document literal Grid service calls, functions to create configurable service properties, functions to specify resource properties and register metadata, functions to specify the security configurations of the service operations of the service, and support to deploy a service to commonly used Grid service containers. The Introduce GDE can be used to create, modify, and deploy a grid service. It is designed to be very simple to use, enable using community accepted data types, and provide easy configuration of service metadata, operations, resources, and security. The Introduce GDE contains several screens and options for a service developer to 1) create a new service, 2) modify an existing service, 3) discover and use published data types in order to create strongly-typed service methods, and 4) deploy the service.
- It enables development of strongly-typed services. As we discussed in Section 2.1, one of the key characteristics of caGrid is the support for interoperable services. The Introduce Toolkit implements this support by enabling creation of services that consume and produce data objects (or data elements), whose XML serialization conforms to XML schemas – the XML schemas may be registered in the GME service of caGrid. These services are referred to as strongly-typed services, since each input and output parameter of the service methods are objects with well-defined structures. Using the Introduce GDE, a developer can 1) import

object/data types for use in a service -- the service developer can use the GME Grid service to import object/data definitions (XML schemas) into the tool or read object type definitions (as XML schemas) from a file system. Introduce provides plug-ins for developers to incorporate other object/data type providers.

- It is customizable and extensible via the use of extension plug-ins. Plug-ins allow for Introduce to be customized and its base functionality to be extended 1) for custom and common service types in an application domain and 2) to employ customized discovery mechanisms for common object/data types for creating strongly-typed services.
- It allows for implementation of secure services. It leverages the caGrid security infrastructure, GAARDS, in order to provide customizable service- and method-level security configuration and code generation. It provides support for service developers to optionally turn on authentication and authorization support for individual service methods as well as the entire service itself. When configured, the service can interact with Grid-wide and local authentication and authorization services to enforce controlled access to its methods.
- It manages all the service-specific files and directories required by the GT for correct compilation and deployment. It also generates appropriate, object-oriented client APIs which can be used by client applications to interact with the service.

For a Grid service developer, Introduce is the primary tool used to create new Grid services and modify existing Grid services. There are four primary functions performed in Introduce: 1) Optionally choose service extensions to customize the type of Grid service, such as the Data Service extension to create a data service, or the caGrid Transfer extension to support large data transfer; 2) import data types (object definitions) for use in the Grid service; 3) define Grid service operations, specifying inputs, outputs, and operation faults; define service properties (configurable properties that the service uses to customize run-time behavior); 4) if the service is a data service, choose the CQL query processor and customize query processor properties; and 5) set service security options, including authentication and authorization policies.

**Choosing Service Extensions.** Introduce extensions are additional pre-defined service extensions that add additional capability to a Grid service. An example is the Data Service extension. By adding a data service extension to a Grid service, the Grid service becomes a “Data Service”. The extension adds the following: 1) object types related to the caGrid query language, called CQL, including the CQLQuery object and the CQLQueryResultCollection object; 2) a standard service operation, “query”, that takes a CQLQuery object as input and returns a CQLQueryResultCollection object; 3) a query processor that transforms a CQL query into the query language supported by the backing data store (e.g., HQL, Hibernate Query Language). The service developer can customize the query processor properties. Example properties include configuration of the backing data store where data resides. There are additional optional features the Data Service extension supports that a service developer can customize, such as enabling data service auditing or enabling CQL validation (validating CQL against the data service’s domain model).

**Importing Data Types.** Object/data types are imported into the service so that the service developer can re-use existing object/data types. For example, consider the use case of developing an analytical service processing Grid Gene objects. The service developer could import the Gene object class (already defined and registered in the caDSR, EVS, and GME) for use in the analytical service.

**Defining Grid Service Operations.** The service developer adds operations to the service. Service operations are analogous to object methods in object oriented programming. Input and output types are chosen from previously imported data types. In addition, the service developer specifies faults that can be thrown by the service.

An advanced activity is to create a stateful Grid service. To create a stateful Grid service, the service developers creates one or more service contexts. These contexts are created by the Grid service when a client invoked a service operation that returns a service context. These contexts have their own operations, and an associated resource that holds state specifically for the client that invoked the Grid service. An example application of a stateful Grid service is a job invocation pattern. In this pattern, the user calls a service operation to create a new job. The service creates a context for the user, and an associated resource, and sets the job parameters on the resource. Then the service returns the context handle to the client. From this point onward, the client invokes operations on the service context to start the job, check status, etc.

**Setting Authentication and Authorization Options.** If the service is “secure”, meaning accessible only to users who have successfully authenticated on the Grid, the Grid service developer sets the appropriate service authentication requirements. For example, the developer can enable “Transport Layer Security”, indicating a private communication method. This configuration option requires users to connect to the service using only the “https” protocol. In addition, the service developer can configure authorization policies. For example, the service developer can customize the Grid Grouper authorization policy for invoking a particular service method: restrict access to service operation X to members of Group Y.

When a user saves the Grid service, Introduce generates a service skeleton. A software developer implements the methods generated by Introduce to enable service functionality. Currently, the language binding supported by Introduce is Java. Introduce generates a complete Java project familiar to Java developers: 1) Eclipse project files; 2) Ant build and deployment targets; 3) service skeleton (source code for which a programmer provides the implementation of service operations); and 4) client classes and APIs for accessing the service.

**Related caGrid Services and APIs.** Because Introduce is a high-level toolkit for creating Grid services, it leverages many other services and APIs comprising caGrid. Some of the major APIs and services used in Introduce include: Data Service extension, caGrid Transfer extension, GME (used to import Grid object definitions), Grid Grouper, caDSR (used to import domain models for data services), and the Index Service (for service registration and metadata advertisement).

### 3.2.caGrid Query Language

caGrid Query Language (CQL) is a custom object-oriented query language. CQL provides a common query language for all data services deployed to the Grid. That is, all Grid queries are expressed in CQL and each caGrid-compliant data service is required to be able to consume CQL queries. CQL is designed to be simple so that service developers can easily implement specialized query processors for different types of backend databases. Use of a common query language across all data services in the environment facilitates federated query of multiple services. caGrid provides a federated query processing service. Federated Query Processor (FQP) takes DCQL (an extended form of CQL) as input to perform federated query. DCQL queries are broken into composite CQL queries and passed to individual data services.

The essence of CQL is the following. First, a user creating a CQL query decides which class in the model they want to retrieve. Then the user specifies restrictions indicating which instances of that class (representing actual data stored in the backing datastore) the user wants to receive. For example, the user can target a “Person” class and specify a restriction requiring the requiring the “age” attribute to equal “5”. CQL supports association traversal, so the user can compose a query that will restrict results based upon attributes in associated objects. An example is requiring the “Institution” class associated with the Person have a name of “The Ohio State University”. CQL was created to query the object model itself. The implementation takes care of performing a query on the back-end data store, which is hidden from the Grid. For additional examples and further details of CQL, please see the developer wiki at <http://www.cagrid.org>.

caGrid provides higher level APIs (with Java binding) to create and submit queries. On the wire, a CQL query is expressed in an XML document conforming to a well defined schema with the URI <http://CQL.caBIG/1/gov.nih.nci.cagrid.CQLQuery>.

### 3.3.Types of Grid Services

There are two main types of caGrid services provided by service providers: Data Services for sharing data in the Grid environment, and Analytical Services that provide access to analysis routines over the Grid.

#### 3.3.1. Data Services

Data Services share data on the Grid. The data might reside in a data repository such as a relational database (RDBMS), XML database, or file system. There are some common features of all caGrid data services: 1) Each data service exposes its data using a well-defined object-oriented domain model. From

a client's point of view, data from the data service is accessed as objects. 2) Each data service has the standard "Query" service interface that takes a CQL query object as input and returns a result collection (of type CQLQueryResultCollection), containing objects that satisfy the query and whose classes are defined in the domain model. 3) Each data service implements a query processor that transforms a CQL query into the query language supported by the backing data store (e.g., HQL, Hibernate Query Language, for caCORE Data Services). The job of a data service implementer is to provide this mapping of database data to objects defined in the domain model.

A common type of data service in caBIG™ is a "caCORE-backed" data service. This is a data service for which the back-end implementation is created by the caCORE SDK (<http://ncicb.nci.nih.gov/infrastructure/cacoresdk>). As with all data services, a domain model which implements an object-oriented view of the backend data source is developed using the Unified Modeling Language (UML). The corresponding UML document (in XMI format) is the primary artifact used as input to the caCORE SDK. The caCORE SDK produces artifacts that are then used to create a caGrid Data Service using Introduce. caGrid offers a data service extension that implements a query processor that translates CQL queries into hibernate select statements. Because the caCORE query processor exists, the requirement to implement a mapping of objects in the UML domain model to the database is already completed. Thus, the process of creating a Grid data service using caCORE SDK is relatively simple. To retrieve large data sets, caGrid implements WS-Enumeration support. This web services specification defines web service interfaces to support enumeration (iteration) over objects. It is essentially a web services (distributed) version of a Java Iterator.

### 3.3.2. Analytical Services

Analytical Services are services that provide access to analysis routines over the Grid. The following are requirements for analytical services:

1. Object-oriented client APIs
2. Strongly typed interfaces

Analytical services consume and produce objects, whose class definitions are well-defined. Service developers can use the Introduce Toolkit to create Analytical Services. In Introduce, object definitions can be imported from the caDSR, or imported from the GME, or created directly from an XMI file (e.g., exported from Enterprise Architect, a popular UML modeling tool).

### 3.4.Security

Security is of primary importance for all Grid participants. Security helps to protect both intellectual property and patient information shared on the Grid. Security concerns addressed by GAARDS include the following:

#### 3.4.1. Grid Account Management and Federation

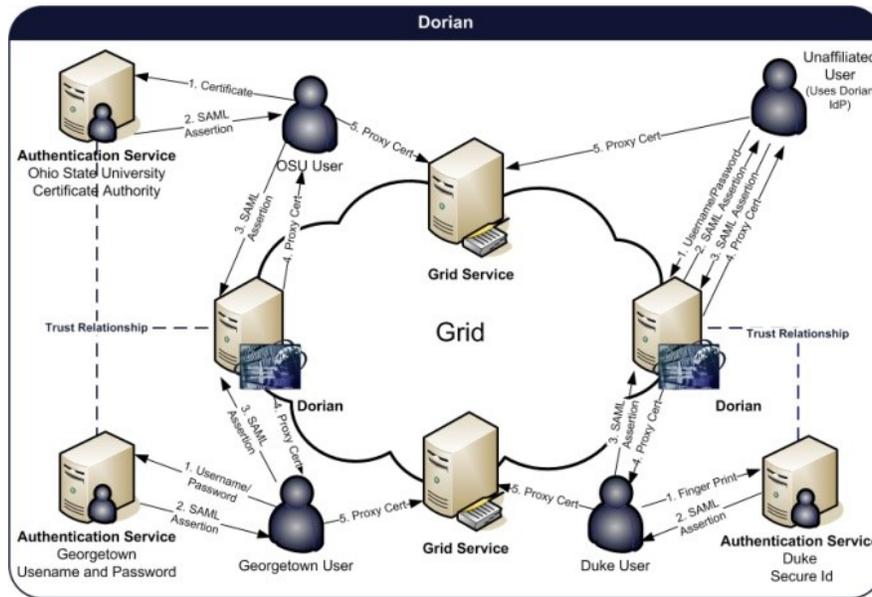
The Globus Toolkit supports security via Grid Security Infrastructure (GSI). GSI supports Grid identities in the form of X.509 certificates. X.509 certificates are an implementation of PKI ([http://en.wikipedia.org/wiki/Public\\_key\\_infrastructure](http://en.wikipedia.org/wiki/Public_key_infrastructure)). The public certificate includes both the identity of the Certificate Authority (CA) that issued the certificate and the user's public key. When a user presents the certificate to a Grid service, the Grid service can use the public key embedded in the certificate to authenticate the user. The user keeps their private key secret, and uses their key to decode/encode messages during communication.

X.509 certificates are used by both individuals (Grid users) and Grid services to identify themselves. caGrid builds upon X.509 certificates to meet Grid requirements. For example, Grid users typically log in from multiple locations. Requiring a user to have their private key and certificate at each location is both cumbersome and a security risk (due to the existence of multiple copies of the private key). caGrid uses proxy certificates to alleviate this need. A proxy certificate is a time-limited (a maximum of 12 hours) certificate that a user can present as identification when invoking Grid services. As long as the service trusts the certificate authority that issued the user's proxy certificate, authentication will succeed. Similarly, the user validates the identity of the remote Grid service by performing the same CA check in reverse.

Using existing tools, the provisioning of Grid credentials is done manually, which is far too complicated for users. The overall process is further complicated in the case where users wish to authenticate from multiple locations, because a copy of the users' private keys and certificates have to be present at every location. Securely distributing private keys is error prone and poses a security risk. Additionally, there are scalability and efficiency problems with vetting user identities. Organizations invest a significant amount of resources into their existing identity management systems and already have processes in place for vetting user identities. In such settings, it would be more efficient to leverage existing identity management systems to provision Grid user accounts. Users would be able to use their existing credentials to "logon" to obtain Grid credentials and access Grid services. This scenario requires a mechanism to allow users to obtain Grid credentials using their existing organization-provided credentials. The mechanism should also remove the complications of using and managing Grid credentials.

## caGrid Technical Overview

Dorian is a Grid user management service that 1) hides the complexities of creating and managing Grid credentials from users and 2) provides a mechanism for users to authenticate using their institution's authentication mechanism. Dorian implements 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 X.509 identity certificates to support user authentication. Dorian uses SAML authentication assertions as the enabling mechanism for federating users from local institutions to the Grid.



**Figure 4.** Dorian system.

Figure 4 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. After 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 will only issue Grid credentials if the SAML assertion is signed by a Trusted Identity Provider. For example, in Figure 4, a Georgetown user wishes to invoke a Grid service that requires Grid credentials. She supplies the application with her username and password. The application client authenticates the user with the Georgetown Authentication Service, receiving a signed SAML assertion which it subsequently passes to Dorian in exchange for Grid credentials. These credentials can then be used to invoke Grid services. To facilitate smaller groups or institutions without an existing identity provider (IdP), Dorian also has its own internal IdP. This allows users to authenticate to Dorian directly. This scenario is also illustrated in Figure 4.

### 3.4.2. Authorization: Grid Service Access Control

The Grid Grouper is a group/virtual organization management solution for the Grid supporting group-based authorization. Grid services and applications enforce authorization policy based on membership to groups defined and managed at the Grid level. The Grid Grouper is built on top of Grouper, which is an Internet2 initiative focused on providing tools for group management. Grouper is a Java object model which 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; delegation. Applications interact with Grouper by embedding Grouper's Java object model inside the application. The Grid Grouper is a Grid-enabled version of Grouper. It provides a service interface to the underlying Grouper object model. Groups are then available and manageable to applications and other services in the Grid. The 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 and enforce a group-membership authorization policy.

In Grouper/Grid Grouper, groups are organized into namespaces called stems. Each stem can have a set of child stems and set of child groups, with exception of the root stem which cannot have any child groups. For example, let's take a university comprised of many departments each of which has Faculty, Staff, and Students. To organize the university in the Grid Grouper, a stem would be created for each department. Each department stem would contain three groups: Faculty, Staff, and Students.

Common Security Module (CSM), developed by the caCORE SDK team, supports permissions in a way that is very similar to standard Unix permissions (read/write access to data elements). Authorization is determined by comparison to a specified permission on the protected structure (similar to how Unix determines read or write access to a file by comparing the user name to the name of the user who owns a file).

A key feature distinguishing CSM from Grid Grouper is the following. CSM centralizes permissions while Grid Grouper is based upon decentralized permissions. That is, a CSM instance is closely tied to the service, database, etc. for which it stores permissions. Grid Grouper is based upon group management; therefore Grid Grouper maintains groups and members of a group, but various institutions are able to create their own groups and manage group members themselves. Typically, one institution manages CSM permissions for the protected resource.

### 3.4.3. Grid Trust Management

Multiple certificate authorities are used on a Grid. Thus, there is a need to manage which CAs are trusted and at what “level of assurance” (that is, how well they are trusted). In a Grid environment, the number of certificate authorities and the number of user identities can grow to be very large. Moreover, 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 will frequently publish Certificate Revocation Lists (CRL), which specify “blacklisted” certificates that the authority once issued but no longer accredits. For the security and integrity of the Grid, it is critical to both authenticate and validate a given credential against an accurate list of trusted certificate authorities and their corresponding CRLs. The Grid Trust Service (GTS) is a federated infrastructure enabling the provisioning and management of a Grid trust fabric. The salient features of GTS are as follows:

- A complete Grid-enabled federated solution for registering and managing certificate authority certificates and CRLs, facilitating the enforcement of the most recent trust agreements.
- Definition and management of levels of assurance, such that certificate authorities may be grouped and discovered by the level of assurance that is acceptable to the consumer.
- Due to the federated nature of GTS and its ability to create and manage arbitrary arrangements of authorities by level of assurance, it facilitates the curation of numerous independent trust overlays across the same physical Grid.
- Client validation, allowing a client to submit a certificate and trust requirements in exchange for a validation decision, which allows for centralized certificate verification and validation.

### 3.4.4. Credential Delegation Service (CDS)

The CDS is used by a Grid user to securely and temporarily hand their credentials to a target service (or another user) to allow the service to perform work on their behalf. A primary use case for credential delegation is workflow. The brief explanation of how this works is as follows. A user logs on to the Grid to retrieve their credentials. The user then wants to execute an analysis workflow which requires access to two data services and a chain of analytical services to process the data from the two data services. The user composes the workflow and submits it to the workflow management service of caGrid. In this scenario, the user needs the workflow management service to have his Grid credential to perform work on his behalf. For example, the workflow management service needs to submit a query to the two data services and invoke the first service in the processing chain with the results of the query, and then invoke the second service in the chain with the output from the first service, and so on. The workflow management service will need to be handed the user’s credentials to access these services. The user

delegates his credentials to the CDS, specifying that the workflow management service (identified by using the service's Grid identity) can retrieve the user's credentials for a limited period of time. Once the delegation is complete, the user hands a reference to the delegated credential to the workflow service. The workflow service uses the reference to actually retrieve the user's Grid credential and run the analysis.

Introduce is the primary tool that leverages GAARDS to enable authentication and authorization on Grid services. In addition, the GAARDS Security UI is used by Grid administrators to: provision accounts; manage the trust fabric; create and manage Grid Grouper groups; and manage host certificates issued by Dorian.

### 3.5. Large Data Transfer

caGrid supports transferring large amounts of data around the Grid. Currently, there are two options: 1) GridFTP ([http://www.globus.org/grid\\_software/data/gridftp.php](http://www.globus.org/grid_software/data/gridftp.php)) and 2) the caGrid Transfer service. GridFTP is a high-performance FTP server extended to support authentication using X.509 certificates. It is both robust and offers excellent performance, but its installation is difficult (e.g., there is no Windows version of GridFTP). The Transfer service included in caGrid is much simpler to use and offers excellent performance. Transfer supports WS-Notification, allowing interested parties to receive updates on the status of data transfers. The Introduce Toolkit offers a caGrid Transfer extension that leverages the Transfer service. Introduce also offers a Bulk Data Transfer (BDT) extension. This extension adds a few service methods and a service context to allow a service developer to support large data transfer by utilizing GridFTP or caGrid Transfer in their service implementation.

### 3.6. Web Integration

There are two pieces to web integration in caGrid: 1) integration between the web browser security model and the Grid security model, and 2) integration between portlets and Grid services.

#### 3.6.1. WebSSO

Web Single Sign-On (webSSO) is the caGrid product that supports integrating Grid sign-on into a web browser portal. The current implementation targets the Liferay portal (<http://www.liferay.com/>) platform. The webSSO design implements the single sign-on concept: 1) a user logs into the portal, which redirects the user to an external login mechanism (JA-SIG's Central Authentication Service, CAS), e.g., Dorian; 2) after login, the webSSO implementation delegates credentials to the Credential Delegation Service; and 3) each portlet can hand the credential reference to services that the portlet

invokes, or alternatively use the credential retrieved from CDS to directly invoke grid services on behalf of the user. In summary, webSSO integrates Grid sign-on with Liferay portal sign-on to provide a seamless user experience.

### 3.6.2. caGrid Portal

The caGrid Portal (deployed at <http://cagrid-portal.nci.nih.gov/>) is a web portal that leverages many of the caGrid components presented in this overview. It is an excellent demonstration of the capabilities of the Grid.

As a brief walkthrough, Grid services, which have registered with the Index service and advertised appropriate metadata, are displayed on a U.S.A. map on the front page of the portal. Filtering can be performed both by service type (data/analytical services) and by organization offering the Grid service. Once data services have been identified, the portal can be used to query the service, retrieving results as XML. Note that concepts can be used when searching for compatible services (e.g., all services taking a Gene concept as input). Federated queries can be performed across services. The caGrid Portal uses most of the services and APIs in caGrid to provide a user interface offering: Grid service discovery; browsing of metadata; CQL query formation and execution; and more.

## 3.7.caGrid Coordination Services

### 3.7.1. Index Service and Metadata

The Index service is the white and yellow pages of the Grid. All services participating in a Grid should, but are not technically required to, advertise to the Index Service. Typically, there is one Index service per Grid. However, Index services can be linked to form a federation of Index Services. For the purposes of Advertisement and Discovery, caGrid leverages the Globus-provided Index Service. The Index Service implements the standard WS-ServiceGroup specification. When services are added to the service group, they specify what and how metadata should be accessed from them, and the Index Service performs this aggregation. Clients can then query this aggregated information using standard Resource Property operations. caGrid services are expected to maintain soft-state registration to a well-known, Index Service instance, specifying polling of standard caGrid standard service metadata. Traversing (querying) an Index Service is performed via XPath query. Interested parties can subscribe to changes in the Index Service contents. An example subscription would be notification of new services advertising to the Index service. For more information on the Index Service, see the Globus documentation (<http://www.globus.org/toolkit/docs/4.0/info/>).

Service metadata typically advertises information about the deployed service. For example, what organization deployed the service, where the service is located, etc. Service metadata is represented as XML values stored as simply <key, value> pairs. Standardization and agreement upon what service metadata will be advertised is a key step in deploying services to the Grid. The common service metadata contains information about the service-providing cancer center, such as the point of contact for the service and the institution's name which is providing the service. Data Services provide an additional "domain model" metadata, which details the domain model, including associations and inheritance information, from which the objects being exposed by the service are drawn. These metadata standards leverage the data models registered in caDSR and link them to the underlying semantic concepts registered in EVS. The common service metadata for analytical services details the objects, used as input and output of the services operations, using the same format as the data service metadata. In this way, all services fully define the domain objects they expose by referencing the data model registered in caDSR, and identify their underlying semantic concepts by referencing the information in EVS. caGrid provides a series of high-level APIs and user applications for performing lookup on service metadata which greatly facilitate the discovery process. A Grid user can query the Grid to find services that, for example, use a Gene object as input. As an additional step, service metadata publishes the concepts used in the service to support service discovery by concept. The advertisement and discovery process is illustrated in Figure 2. A practical application of using service metadata is the caGrid portal: <http://cagrid-portal.nci.nih.gov/>.

**Related caGrid Services and APIs.** Metadata is set using the Introduce toolkit. Metadata is published to the Index Service. The Discovery API can be used to query metadata registered in the Index Service to perform semantic search for Grid resources. The Index service is closely tied to metadata, the caDSR, and GME. The Index Service aggregates information from other services to support comprehensive semantic discovery on the Grid.

### 3.7.2. Federated Query and Workflow

caGrid provides a workflow management service that supports the execution and monitoring of workflows expressed in the Business Process Execution Language (BPEL). WS-BPEL is the current standard for specification of workflows in service-oriented infrastructures. Using this standard in the middleware infrastructure facilitates easier sharing and exchange of workflows, an important feature of collaborative environments. There are already several BPEL editors available, however, a low abstraction level of such interface prevent their use by end users. Recent efforts have been made to integrate with the Taverna Workflow Management System (in version 1.2 of caGrid core infrastructure) as part of a collaboration with the Integrative Cancer Research Workspace of caBIG™. For more information about composing and executing workflows in caGrid using Taverna, please refer to: [http://www.cagrid.org/wiki/CaGrid:How-To:Create\\_CaGrid\\_Workflow\\_Using\\_Taverna](http://www.cagrid.org/wiki/CaGrid:How-To:Create_CaGrid_Workflow_Using_Taverna).

The caGrid infrastructure provides support for federated querying of multiple data services to enable distributed aggregation and joins on object classes and object associations defined in domain object models. The current support for federated query is aimed at the basic functionality required for data subsetting and integration. The Federated Query Processor (FQP) service provides support for joins queries over multiple data services. The FQP service consumes a Distributed CQL (DCQL) query (an extension of CQL with additional language constructs specifically supporting joins) and splits the query up into standard CQL queries assigned to multiple data services. The results from each data service query are aggregated and joined in the FQP, and the entire result set is returned to the requester.

**Related caGrid Services and APIs.** FQP is closely related to CQL, DCQL. FQP is more loosely related to data services, because a federated query is executed against two or more Grid data services.

### 3.7.3. Global Model Exchange (GME) Service

This service supports storage and retrieval of XML schemas. These schemas define the wire representation of abstract types (corresponding to UML class definitions) stored in the caDSR. The GME categorizes schemas by namespace. The convention used for namespaces is the following: `gme://<Classification Scheme>.<Context>/<Scheme Version>/<Scheme Item>`. “<Classification Scheme> defines the project (or application; e.g., RProteomics, caBIO) within the <Context> (e.g., caBIG). The version of the schema is encoded in <Scheme Version> section of the namespace. The name or id of the schema is stored in the <Scheme Item> section.”

**Related caGrid Services and APIs.** The GME stores the wire definition of object types. The GME is used by Introduce to import Grid object definitions.

### 3.7.4. caGrid Services for caDSR and EVS

The caGrid caDSR Grid Service provides access to information in the caDSR that is relevant to caGrid, and has capabilities to generate caGrid standard metadata instances. Specifically, the service provides operations to access UML-like information stored in the caDSR. It also has operations to generate Data Service metadata for a described subset of a given project registered in caDSR. Finally, it has an operation which augments a description of an Analytical Service, via a partially populated service metadata instance, with the necessary UML-like and semantic information, extracted from caDSR, to describe the service and its operations. The EVS caGrid service allows programs to utilize the caGrid Infrastructure to access EVS information that is currently being produced by NCICB.

**Related caGrid Services and APIs.** The caDSR is closely related to the domain model defining Grid objects. Annotations on the UML model are used to populate the caDSR. caDSR is also related to the Index Service and to Introduce, since it stores the semantics of Grid types. EVS defines the concepts that

are used to annotate domain models. By annotating a UML domain model with EVS concepts, the model designer is explicitly linking each class and attribute to a concept to indicate the meaning of the entity.

### References

- [1] S. Oster, Hastings, S., Langella, S., Ervin, D., Madduri, R., Kurc, T., Siebenlist, F., Foster, I., Shanbhag, K., Covitz, P., Saltz, J., "caGrid 1.0: A Grid Enterprise Architecture for Cancer Research," in *Proceedings of the 2007 American Medical Informatics Association (AMIA) Annual Symposium* Chicago, IL, 2007.
- [2] S. Oster, S. Langella, S. Hastings, D. Ervin, R. Madduri, J. Phillips, T. Kurc, F. Siebenlist, P. Covitz, K. Shanbhag, I. Foster, and J. Saltz, "caGrid 1.0: An Enterprise Grid Infrastructure for Biomedical Research," *Journal of American Medical Informatics Association (JAMIA)*, vol. 15, pp. 138-149, 2008.
- [3] J. Saltz, S. Oster, S. Hastings, S. Langella, 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*, vol. 22, pp. 1910-1916, 2006.
- [4] F. Berman, A. J. Hey, and G. Fox, "Grid Computing: Making The Global Infrastructure a Reality," John Wiley & Sons, 2003.
- [5] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," *Journal of Computational Science and Technology*, vol. 21, pp. 523-530, 2006.
- [6] I. Foster, S. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International J. Supercomputer Applications*, vol. 15, 2001.
- [7] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," San Francisco: Morgan Kaufmann, 1999.
- [8] 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.
- [9] 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.
- [10] 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.," 2004.
- [11] I. Foster, K. Czajkowski, D. Ferguson, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke, "Modeling and Managing State in Distributed Systems: The Role of OGSF and WSRF," *Proceedings of IEEE*, vol. 93, pp. 604-612, 2005.
- [12] 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.
- [13] P. A. Covitz, F. Hartel, C. Schaefer, S. Coronado, G. Fragoso, H. Sahni, S. Gustafson, and K. H. Buetow, "caCORE: A Common Infrastructure for Cancer Informatics," *Bioinformatics*, vol. 19, pp. 2404-2412, 2003.
- [14] J. Phillips, R. Chilukuri, G. Fragoso, D. Warzel, and P. A. Covitz, "The caCORE Software Development Kit: Streamlining construction of interoperable biomedical information services.," *BMC Medical Informatics and Decision Making*, vol. 6, 2006.

- [15] S. Langella, S. Oster, S. Hastings, S. Siebenlist, F. Phillips, J., Ervin, D., Permar, J., Kurc, T., Saltz, J., "The Cancer Biomedical Informatics Grid (caBIG™) Security Infrastructure," in *Proceedings of the 2007 American Medical Informatics Association (AMIA) Annual Symposium* Chicago, IL, 2007.
- [16] S. Langella, S. Oster, S. Hastings, F. Siebenlist, T. Kurc, and J. Saltz, "Dorian: Grid Service Infrastructure for Identity Management and Federation," in *The 19th IEEE Symposium on Computer-Based Medical Systems, Special Track: Grids for Biomedical Informatics*, Salt Lake City, Utah., 2006.
- [17] S. Langella, S. Oster, S. Hastings, F. Siebenlist, T. Kurc, and J. Saltz, "Enabling the Provisioning and Management of a Federated Grid Trust Fabric," *6th Annual PKI R&D Workshop, Gaithersburg, MD*, April 2007.
- [18] S. Langella, S. Hastings, S. Oster, T. Pan, A. Sharma, J. Permar, D. Ervin, B. Cambazoglu, T. Kurc, and J. Saltz, "Sharing Data and Analytical Resources Securely in a Biomedical Research Grid Environment " *Journal of the American Medical Informatics Association* vol. 15, pp. 363-373, 2008.
- [19] S. Hastings, S. Oster, S. Langella, D. Ervin, T. Kurc, and J. Saltz, "Introduce: An Open Source Toolkit for Rapid Development of Strongly Typed Grid Services," *Journal of Grid Computing*, vol. 5, pp. 407-427, 2007.