National Cancer Institute Center for Bioinformatics (NCICB)

caGRID White Paper

(Cancer Biomedical Informatics Grid Prototype Project)

July 23, 2004

William Sanchez [1]
Brian Gilman [2]
Manav Kher [1]
Steven Lagou [2]
Peter Covitz [3]

[1] Science Application International Corporation (SAIC)
[2] Panther Informatics
[3] NCI Center for Bioinformatics

caBIG cancer Biomedical Informatics Grid

TABLE OF CONTENTS

1 Abstract

The lack of common infrastructure has prevented life science research institutions from being able to mine and analyze disparate data sources. Research facilities have been working with islands of isolated data and informatics tools, a situation that impedes the dissemination, aggregation and reuse of scientific data and information. The inability to share technologies and data developed by different cancer research institutions can severely hamper the research process. The lack of unifying architecture is a major roadblock to a researcher's ability to mine different databases. The cancer Biomedical Informatics Grid (caBIG) project was created to provide the organizational framework that could tackle these issues in a "big" way.

In anticipation of the needs that would emerge in the caBIG program, the NCI Center for Bioinformatics (NCICB) evaluated the current state of a number of technologies that have been developed to facilitate data and system integration across wide area networks. The Globus Tool Kit and the Open Grid Services Architecture-Data Access Integration (OGSA-DAI) were selected as the basis for the development of a prototype system that satisfied simple data integration and sharing use cases.

In the proof of concept presented here, it was shown that two disparate physically independent data sources were shown to be mapped into the same object representation. The OGSA-DAI framework was extended to support additional metadata and to connect to the NCI's cancer Bioinformatics Infrastructure Objects (caBIO) application programming interfaces (APIs), rather than to a relational database. Further, the data sources were shown to be accessible through a common interface that enabled data sharing. The prototype was dubbed "caGRID," and the lessons learned may inform the ultimate system architectural designs for caBIG.

2 Background

Biomedical research has entered a new phase. The completion of the Human Genome Project sparked the development of many new tools for today's biomedical researcher to use in finding the mechanism behind disease. Coupled with the sequencing and annotation of many model organisms, our ability to risk-stratify patients using a collection of phenotypic and genotypic information may come to fruition in the foreseeable future. Our ability to deduce correct dosage, drug efficacy, and provide pre-clinical intervention for at risk patients may become a reality.

While the goal is clear, the path to such discoveries has been fraught with roadblocks in terms of technical, scientific, and sociological challenges. The deluge of data that large-scale sequencing, transcriptomic and proteomic studies have produced to date is a case in point. In addition to the shear volume, data collected using a variety of laboratory technologies and techniques are often published without the background information (method of capture, sample preparation, statistical techniques applied) that is needed to reproduce results. In fact, a typical researcher spends as much time trying to understand the origins of a dataset as actually performing new analyses.

The situation is even more problematic in the clinical research domain, where data collection is still often performed on paper forms that differ from study to study, even when the same types of data are being collected. Rarely is a clinical biostatistician able to make good use of data collected on studies they were not directly involved with, largely due to incomplete or non-existent annotation and standardization of the information.

This data problem has pushed the biological community to partition and compartmentalize their data for easy digestion and maintenance. While this approach worked in the past for simple systems containing a relatively small number of interactions, modeled by a small number of datasets, bioinformaticians are finding it difficult to model more complex systems. The simplicity and digestibility of the compartments described above have made it almost completely impossible to cross compartmental boundaries without consulting an expert.

To alleviate this burden, bioinformaticians are starting to apply sophisticated computational approaches in the areas of statistics, data mining, signal processing and artificial intelligence to discover relationships between such compartments. However, the community is quickly discovering glaring inconsistencies in language, methodology and computational models used to describe a particular organism, pathway, interaction, annotation, and so forth. The repercussions of the compartmental approach have produced a bottleneck in the road to discovery. To make this more concrete, computational approaches to data analysis and discovery typically rely on formalism in terms of syntax, context, and format in order to perform reproducible and consistent experiments (the backbone of hypothesis driven science). These formal definitions are severely lacking in the biological sciences. They will remain a burden to the process of biological discovery unless the biological community takes action.

2.1 caCORE Approach

The caGRID project aims to produce a software system capable of handling biological data that crosses compartments. The approach is to leverage existing technologies for the standardization of data representation and semantics and couple them to a common data integration architecture. caCORE is NCICB's platform for data management and semantic integration, built using formal techniques from the software engineering and computer science communities. caCORE defines a data model specified using industry standard techniques to define common biological constructs (objects). By mapping compartments to these common data objects, the process of data integration can commence. More importantly, activation energy needed to cross and interoperate amongst compartments is lowered enough to allow for more sophisticated biological discovery.

The main components of caCORE include:

- **Cancer Bioinformatics Infrastructure Objects (caBIO):** Biological middleware facilitating multi-disciplinary data integration and software re-use through platform independent APIs that reflect an object-oriented view of biomedical information.
- **Cancer Data Standards Repository (caDSR):** A metadata registry based upon the ISO/IEC11179 standard that is used to register the descriptive information needed to render cancer research data reusable and interoperable. caBIO data classes are registered in the caDSR, as are the data elements on NCI-sponsored clinical trials case report forms.
- **Enterprise Vocabulary Services (EVS):** Controlled vocabulary resources that support the life sciences domain, implemented in a description logics framework. EVS vocabularies provide the semantic 'raw material' from which data elements, classes and objects are constructed.

These components, to varying degrees, are currently supporting the NCI intramural and extramural informatics and research communities including organizations like the Specialized Programs of Research Excellence (SPORE), Cancer Therapy Evaluation Program (CTEP), Center of Cancer Research (CCR), Division of Cancer Prevention (DCP), and Mouse Models of Human Cancers Consortium (MMHCC).

2.2 caGRID: A Proposal for Interoperability in caBIG

While the caCORE framework has proven to be a successful platform for NCICB-directed informatics projects and systems, it does not provide a mechanism for broader integration across the universe of cancer research institutions nationwide. Any site can set up a caCORE-based system, but there is no formal mechanism for connecting these systems together. Furthermore, there is no simple way to integrate caCORE-based systems with other types of data resources without an additional engineering effort.

For these reasons, NCICB embarked on a project in December of 2003, now dubbed “caGRID”, whose mission was to research, explore, and implement a prototype application to connect disparate data sources together using the best available technology. The outcome and experiences of caGRID, whether successful or not, will be shared with the caBIG community to aid and guide their technology decisions. We wrote this white paper to describe the process used to explore these technologies, our analysis of requirements and architecture, and the prototype grid system itself.

The following sections describe our research and findings. First, we outline our requirements analysis and proceed to describe the technology evaluation and summarize our results. We review the architecture chosen for the prototype, with observations about the places we needed to extend the chosen grid technology to satisfy our requirements. Finally, we summarize our lessons learned and places we would like to see more development. We presented these findings to the caBIG Architecture Workspace that is the body responsible for defining the system and architectural compatibility requirements across the caBIG program.

3 Requirements Analysis

The Cancer research community depends on data located in multiple disparate data sources around the world. In most cases, these data sources do not utilize the same underlying data structure. However, given these tremendous barriers to their research and discovery efforts, the cancer research community has been able to perform a wide range of studies to elucidate the mechanisms and treatment of cancer. Most participants in the cancer community have access to the internet and use protocols such as file transfer protocol (FTP) and hypertext transfer protocol (HTTP) to access disparate data sources (See Figure 3-1 below). As the community grows, many participants have become blind to the seemingly endless number of new datasets, methods and tools published on a daily basis (for example, tools not published through traditional mediums).

We propose a system capable of providing a “yellow pages” facility that identifies the provider, quality, provenance, language and methodology used to capture clinical, genomic, proteomic, and transcript data “on the fly”.

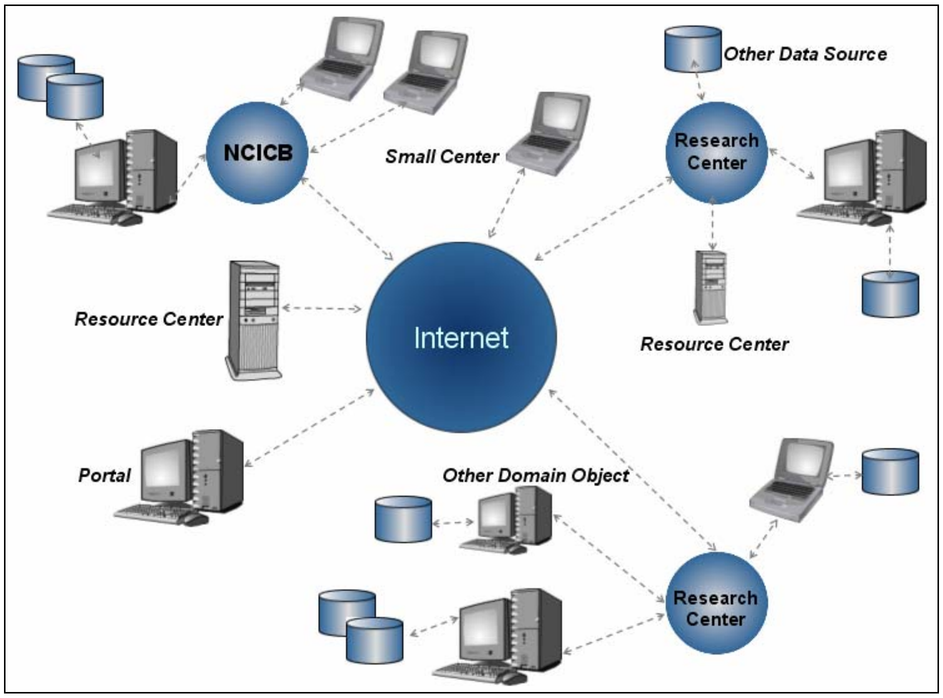


Figure 3-1

In order to facilitate prototype development, we identified caBIO and caCORE tools from NCICB as sources of data that we would expose in the grid.

We identified 12 use cases that serve as the basis for the functional requirements of the grid. We grouped these use cases into three high level categories as follows:

1. **System Administration**

   a. **Startup:** This use case describes the caGRID startup process triggered by either the operating system or the caGRID user. During the caGRID installation, the user can select to start the service automatically or manually. When automatic is selected, caGRID is launched when the system boots up. When manual is selected, caGRID is launched by the user from the command line.

b. **Shutdown:** This use case describes the caGRID shutdown process triggered by either the operating system or the caGRID user. The functionality will be command line.

c. **Configure Data Source and Domain:** This use case describes how to configure caGRID either locally or at the site level. The use case includes configuring new data source and new domains to a local caGRID.

d. **Configure Object to Data Mappings:** This use case describes how to map Object models such as caBIO with a local Data Source at the meta-data level. The mapping process is between one object-attribute from the object model with one field-table from the data source. After the actor maps the data, the system persists the mappings which caGRID uses during local data retrieval. Object to data mapping can be launched from the installation process or from the caGRID admin tool.

e. **Install caGRID wizard:** This use case describes caGRID installation on the user's computer. The user has different options to install caGRID. The simplest option is to download and install the software without starting caGRID.

f. **Setup Manual Domain Object Data Mapping (like NCICB):** This use case describes how an actor performs manual mapping of local data with NCICB data (at the data level). The system persists these data mappings which caGRID uses to submit the query to other Data Source Indicators (DSIs).

g. **Setup EVS Mapping:** This use case describes how an actor performs Symantec mapping of unmatched local data with Enterprise Vocabulary Service (EVS) data. The system persists these data mappings which caGRID uses to submit the query to other DSIs.

h. **Setup Privileges:** This use case describes the configuration and functionality of the security component involved in caGRID. The security identifies who can access what data.

i. **Login Admin Tool:** This use case describes the caGRID login admin tool graphical user interface (GUI) process.

2. **Mapping and Advertisements**

a. **Advertise Services:** This use case describes how caGRID at a given site publishes data services. That is, what domain objects is it going to serve and based on what criteria. The functionality will be command line.

b. **Advertise Data Source:** This use case describes how caGRID at a given site publishes data services. That is, what domain objects is it going to serve and based on what criteria.

3. **Query and Discovery**

a. **Query Data and Discovery Service:** This use case allows the user to retrieve data from multiple Data Source Indicators (DSIs). A DSI refers to the data source at a site. These DSIs are available through Advertisements/Services to other caBIO servers. The actor has the option to query the system in the following different ways: Request data sources (local and remote), request object to data mapping, request attributes, or request data. Using this API, the client will be able to discover all DSIs and the metadata that they support. Discovery is the process of one peer searching for another peer, in the same peer group, that contains the desired content.

We also identified 12 groups of supplementary requirements that are the non-functional requirements. These requirement categories identify the technology environment to implement the project. The 12 groups of supplementary requirements are:

1. Functionality
2. Security
3. Scalability

4. Resource management
5. System properties
6. Group support
7. Monitoring
8. Connectivity
9. Decentralization
10. Event server crash
11. Quality of services
12. NCI standards

Appendix A provides a detailed list of the supplementary requirements and Appendix B provides detailed use case specifications.

4 Technology Evaluation

We developed a preliminary architecture based on the requirements gathered. As shown in Figure 4.1, our architecture identifies four layers - data source, metadata, GRID, and client layer. This architectural view represents a conceptual model that maps the requirements to the technologies chosen.

.

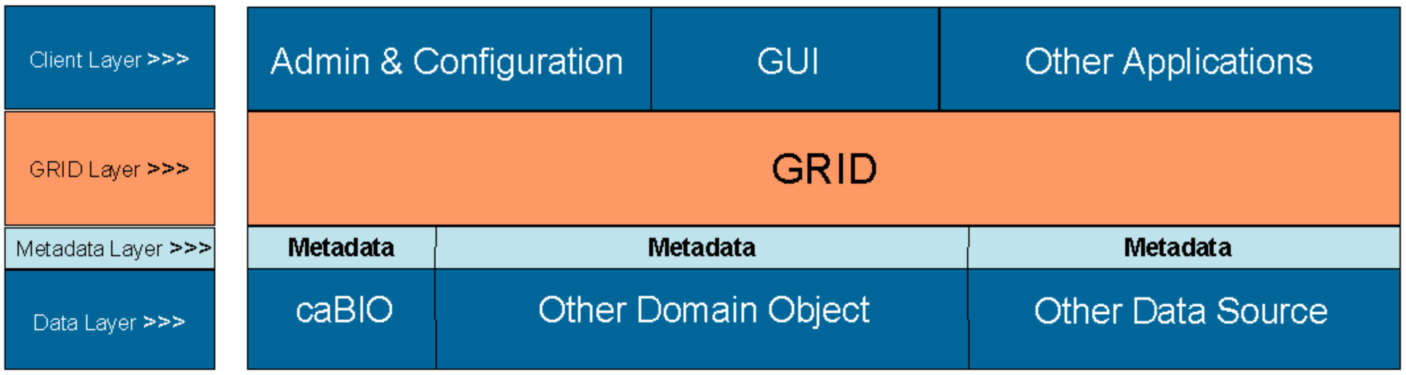


Figure 4.1

Following is a description of each layer:

- **Data source:** This layer represents the different sources that are sharing data in the system. Data can be shared directly from a database, through an API or a file system.
- **Metadata:** We identified two high level uses for metadata in the prototype system:
  - Identification in terms of provenance
  - Identification in terms of semantics
- **Grid:** This layer provides advertisement, discovery and query functionality.
- **Client layer:** This layer provides a programmatic interface to the system. The main purpose of this layer is to provide tools to implement applications. Optionally, the user can interact and manipulate the system resources through a GUI.

Based on this preliminary architecture, the requirements and use cases described above, we sought to enumerate technologies and projects that would satisfy our needs in building the prototype. The next section outlines each technology that we identified, its role in the grid and prototype. It is important to remember that the technological landscape of the grid is changing at a rapid pace. Therefore, it is important to note that we may not have captured all technologies that may have satisfied our requirements today.

4.1 Technologies for Collaboration

We identified web services, grid technologies and peer-to-peer (P2P) as the main emerging technologies for collaboration. The first two are service-oriented architectures (SOA) used to build reliable distributed systems that deliver application functionality as services with the additional emphasis on loose coupling between interacting services.

4.1.1 Technologies Identified

Following are the technologies identified:

- **Web Services:** Web services are software systems identified by a URL, whose public interfaces and bindings are defined and described using XML. Web services definitions can be discovered by other software systems. These systems may then interact with the web service in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols.

- **Grid Computing:** Broadly defined, grid computing is a kind of distributed network computing based on the principle of virtualization of computing and data resources to provide seamless access to vast IT capabilities.

- **P2P:** P2P is the sharing of computing resources and services by means of direct exchange between systems. These resources and services include the exchange of information, process cycles, cache storage, and disk storage for files. This technology takes advantage of existing desktop computing power and network connectivity.

4.1.2 Grid Computing Standards

Web service technology is a standard in the IT industry while grid computing is evolving and maturing. Currently there are different organizations identifying grid requirements and defining grid standards like the Global Grid Forum (GGF), Organization for the Advancement of Structured Information Standards (OASIS), and the World Wide Web Consortium (W3C).

New standards like the Web Service Resource Framework (WS-RF) are evolving to enhance web services by implementing grid service functionality such as stateful resources and resource lifetime.

4.1.3 Related Projects

Many organizations have started identifying the major business areas for grid computing based business applications. Some examples include [15]:

- Life science, for analysis and decoding strings of biological and chemical information
- Financial services, for running long, complex financial models, and arriving at more decisions that are accurate
- Higher education for enabling advance, data and computation intensive research
- Engineering service, including automotive and aerospace, for collaborative design and data intensive testing
- Government, for enabling seamless collaboration and agility in both civil and military departments and other agencies
- Collaborative games for replacing the existing single server online games with more parallel, massively multiplayer online games

The Life science sector has made dramatic advances, providing rapid changes in the way that drug treatment and drug discovery efforts are conducted. The analysis and information technology efforts surrounding genomics, proteomics, and molecular biology efforts have accelerated advancements in computing. These advances have now presented a number of technical hurdles in terms of IT infrastructure required to perform complex computational analysis. This sector is looking toward the grid community for an answer.

Areas of scientific interest that are using grid technology include [16]:

- **Scientific data federation (The worldwide telescope):** The project federated data from hundreds of individual instruments, allowing a new generation of armchair astronomers to perform analyses of unprecedented scope and scale.

- **Medical data federation (Biomedical Informatics Research Network):** The project aimed at federating biomedical imaging data. Biomedical Informatics Research Network (BIRN) is deploying compute-storage clusters at research and clinical sites around the United States and is deploying grid middleware to enable the integration of image data from multiple locations for the proposes of research and, ultimately, improved clinical care.

- **Knowledge integration:** In silico experiments in bioinformatics, it is a different perspective of data federation and analysis problem. The myGRID project focuses on the semantically rich problems of dynamic resource discovery, workflow specification, and distributes query processing, as well as, provenance management, change notification, and personalization.

Examples of life science based grid projects are:

- **BioMOBY:** This project uses web service as integration technology infrastructure. In phase two of the BioMOBY project (not yet implemented), they plan to combine web services and the semantic web.

- **Chinook:** This project uses hybrid implementation using P2P and web service technologies that does not concentrate on data as a service but rather on analytical techniques.

- **DiscoverySpace:** This project provides a graphical user interface for visualization and analysis of DISCOVERYdb, a centralized data warehouse. The software allows the analysis of biological data without the need for the researcher to access multiple data sources and formulate complex Structured Query Language (SQL) queries and scripts. Its primary focus is for interpreting Serial Analysis of Gene Expression (SAGE) data.

4.2 Technologies

In this section, we describe the most significant technologies that support the caGRID requirements. Each technology supports one or more layers from the preliminary architecture described above. We categorized the technologies in the following groups:

- **Web Services**
- **Grid Computing**
- **Peer to Peer**
- **Object Oriented Data Representation**
- **Semantic Web**

4.2.1 Web Services

This technology category represents the web service itself to implement collaboration systems as well as one bioinformatics application implemented over web services.

Web services technology use the HTTP protocol and XML as the base language, making it a very attractive technology for the caGRID project. Universal Description, Discovery, and Integration (UDDI) provides a convenient lookup service for service discovery.

4.2.1.1 Test

Conceptual review was done for the technology stack.

4.2.1.2 Test Results

Web services provide a convenient service oriented framework and this architecture is well suited for distributed system integration. In our evaluation, it was determined that a substantial amount of effort will be needed to implement the caGRID use cases using web services. Web services are semantically poor; both the Advertisement and Discovery use cases cannot be implemented with UDDI, without several coding extensions to provide more explicit metadata. More resources will be needed to implement some kind of grid wide security standard, resource scheduling and other components that are provided by some of the other solutions we tested.

4.2.1.3 BioMOBY

From the Moby website [14]:

"MOBY provides an easy way to create web services, as well as a central registry that has information on all the services that are available. You can use it to create programs that provide data or algorithms to your colleagues over the internet. Your service is associated with a URL so that anyone can access it.

Your data and algorithms are available through the internet but you do not have to create a web page with a form to access it. You are providing an interface that is accessed by programs. If you also want to provide a web form for people to cut/paste data into, you create that separately and use an underlying MOBY service for the data look-up and algorithms.

The BioMOBY project has been split into two branches, MOBY Service Branch (MOBY-S) and Semantic MOBY branch (S-MOBY)

MOBY-S is a web services based approach to data discovery and integration. The main benefit of MOBY-S is that it provides a central registry of services called MOBY-Central, which is similar to your telephone directories:

- White pages: who provides services
- Yellow pages: services organized by classification
- Green pages: how to call each service

MOBY pages: given a specific type of data (e.g. a genbank ID, a keyword, or a sequence), find services that take it as input or generate it as output

By writing and registering your web services with MOBY, you will find that your data and algorithms are automatically categorized and made available to the world. Others can find them through easy searches according to keywords or input and output data types.

S-MOBY is still in the design stage. S-MOBY is a semantic-web based approach, which uses RDF, based technologies (for example, OWL). From the caGRID perspective, S-MOBY provides a better semantic based approach to data service discovery and integration

Plans are to merge MOBY-S and S-MOBY

MOBY provides a simple application-programming interface (API) to provide web services. MOBY hides many of the details of the underlying XML technologies, which makes it easier to write web services. You can convert your older web-based services into MOBY web services or you can start afresh with new web services.

MOBY is particularly well suited for biology because it includes a hierarchy of input/output objects designed for bioinformatics. Already provided are objects for storing sequences, BLAST results and many other common items. With MOBY, you can discover web services that do exactly the type of calculation you want. However, it can only find services that were written to work with the MOBY system.

How does MOBY make use of ontologies?

Suppose you want to find information on the Internet related to a protein named CRK. If you go to your favorite web search engine and search for CRK, you will find a massive amount of information related to the acronym CRK, including everything from "Cable Repair Kit" to "Christian Religious Knowledge."

Your search would be much better if you could attach some metadata to limit your search by saying that CRK refers to a protein. With that type of metadata, you might be able to turn up some web services that produce sequences or do alignments for CRKs.

MOBY helps limit your searches. First of all, MOBY only searches for services that fall within the field of biology. In addition, though, MOBY allows you to say what you want to do with a CRK. Do you want to type CRK as a keyword and return Genbank IDs? Alternatively, would you rather go straight to sequences and then submit them to a BLAST search?

When we talk about ontology in MOBY, we are usually referring to ontology of web services. Our services are classified by what types of input and output objects they take. If we possess a Genbank ID, we can very easily query MOBY to determine what can be done with that ID. Or if we have a Protein or DNA sequence, we can find out what services take that as input and thereby discover BLAST or alignment services."

4.2.1.3.1 Test Environment

There was no test environment. It was a conceptual review.

4.2.1.3.2 Testing Results

A MOBY service satisfies some of the functional requirements as outlined in the Query and Discovery use cases. The current MOBY specification does not provide a complete solution for the advertisement of MOBY services as it requires a central repository for advertisement. This central repository must be maintained by a central authority to maintain a 24 x 7 quality of service.

The installation of a new MOBY service is straight forward as it is dependent on well known and often used MySQL database and PERL programming interfaces. An implementation also exists in Java and depends on the Tomcat servlet container.

The caBIO integration is straight forward as MOBY defines an easy wrapper around many data and computational services. These services can be used and extended to use caBIO interfaces.

BioMOBY uses its own proprietary XML format to exchange semantic, object, and data type information. This is one of two obstacles in adopting MOBY. First, the biological community would have to adopt the MOBY XML specification to exchange information. The second obstacle is its lack of communication with other “grid” projects and use of standards that have been developed specifically for the exchange of data in a grid environment.

S-MOBY, which is the current implementation, provides a stable API and a central registry, but is not semantically rich enough to satisfy all the requirements outlined in Service Discovery use case. In addition, we find the MOBY framework, which is based on the web services technology stack, a little rigid and not as easy to extend as some of the others. MOBY-S will provide ontologically defined data types and service descriptors, which might alleviate some of the above-mentioned problems. Since, MOBY-S is still in its design phase, a complete evaluation is needed.

4.2.2 Grid Computing

Although grid computing is in development and is getting mature, many major academic and industrial research projects have successfully deployed grid enabled applications and projects. Grid computing currently spans such diverse domains as biomedical imaging, astrophysics, bioinformatics, engineering, and multiplayer video gaming.

4.2.2.1 Globus Toolkit 3

Globus Toolkit 3 (GT3) Core offers a run-time environment capable of hosting grid services. The run-time environment mediates between the application and the underlying network, and transport protocol engines. GT3 Core also provides development support including programming models for exposing, and accessing grid service implementations [18].

Globus Toolkit 3 (GT3) is based on a new core infrastructure component compliant with the Open Grid Services Architecture (OGSA) [6] and is an open source implementation of the Open Grid Services Infrastructure (OGSI). The implementation is intended to serve as a proof of concept for OGSI and used as a reference for other implementations. Some efforts have been made to standardize parts of this Java based framework within the Global Grid Forum [19]. GT3 Core also provides development support including programming models for exposing and accessing grid service implementations.

Grid services can be viewed as a standard Web service adapted to the requirements typically found in a grid environment. In order to make this as transparent as possible to grid application developers, GT3 Core implements most of the OGSI functionality on behalf of users and providers of Grid services [18].

4.2.2.2 Test

We installed two Globus components (core and information services) in Linux and Windows platforms. We tested the grid functionality exposing caBIO API as grid services. Using caCORE infrastructure we created grid services to query the caBIO, caDSR and EVS data.

To create the grid services, we installed the caBIO client in the Globus servers. We then wrapped the caBIO API with the Globus client to expose some caBIO APIs as grid services.

We tested the new caBIO grid services using both the client java API and the Globus GUI. The test included several steps - finding services from the registry, receiving grid service factories, instantiating the service, defining grid lifetime and querying caBIO data.

Additionally, with the Information Service module we could aggregate data services (metadata) in one Globus server called the registry. The testing also included the notification services, which allow notifying grid clients about changes in the registry.

4.2.2.2.1 Testing Results

Two Globus modules were tested, core and information services. These fulfill most of functional requirements described in query, discovery and advertisement use cases. The framework allows the creation of wrappers for existing objects and to expose them as grid services.

The two components come with a useful GUI tool that allows testing grid services easily. The core service GUI tool allows checking local or remote Globus installations to check grid service status, to instantiate grid services, to query grid services, to browse WSDL and to get data services among others. Information service component provides a GUI tool to browse and scan aggregated services.

Several command line utilities that are part of the Globus Toolkit help in creating and deploying grid services from Java Interfaces or GWSDL documents.

Most Apache axis tools can be used to configure and deploy services. Dependencies include ant, JDK, Tomcat, Axis, and other jars to configure the Globus environment. Once the required packages are installed properly, the toolkit installation is straightforward. Windows and Linux installation processes are the same. Windows installation brings useful command line utilities to perform most gt3 tasks.

Globus toolkit provides two ways to implement grid services: Implementation by inheritance and implementation by delegation (operation provider). The former approach extends a class called *GridServiceImpl*, which contains all the basic of grid services. Although, we tested caBIO-Globus integration using both approaches, the delegation approach is the best choice to integrate existing java code.

We successfully exposed EVS and caDSR as grid services. Implementation using the Globus toolkit (standalone) involves the creation of wrappers to expose them as grid services. This approach may falter when the underlying code base is changing rapidly.

4.2.2.3 OGSA-DAI

OGSA-DAI is an extension to OGSA designed to support data access and integration. The goal of OGSA-DAI is to provide uniform service interfaces for data access and integration through the grid. The aim is that through OGSA-DAI interfaces, disparate, heterogeneous data sources and resources can be treated as a single logical resource. Moreover, OGSA-DAI will allow these same data resources to be incorporated within an OGSA-compliant architecture. OGSA-DAI grid services themselves will then provide the basic operations that can be used to perform sophisticated operations such as data federation and distributed queries within a virtual organization, hiding concerns such as database driver technology, data formatting techniques and delivery mechanisms from clients.

This goal is to achieve by the provision of an efficient grid-enabled middleware reference implementation of the components required to access and control data sources and resources.

OGSA-DAI is intended to contribute to a future in which it is envisaged that scientists will move away from technical issues such as handling data location, data structure, and data transfer and integration, and instead focus on application-specific data analysis and processing [20].

4.2.2.3.1 Test

We installed OGSA-DAI 3.1 over Globus 3.0 in Windows and Linux platforms. We tested the three main OGSA-DAI components instantiating MySQL and Xindice databases.

4.2.2.3.2 Testing Results

OGSA-DAI has three main components: Data Access & Integration Service Grid Registry (DAISGR), Grid Data Service Factory (GDSF) and Grid Data Service (GDS). These components fulfill most requirements defined in the query, discovery and advertisement use cases, with scenarios that require sharing relational database or XML data.

The DAISGR can register and describe services allowing them to advertise their capabilities to the world. Clients can then search the DAISGR for services that match their requirements.

OGSA-DAI grid service factory provides a service creation facility, creating grid data services, which facilitate access to particular data resources [20]. GDSF are persistent services, their lifetime is tied to that of the web service container. The last component is OGSA-DAI grid data service that provides access to data services.

OGSA-DAI comes with a GUI tool that enables to test the three components. We primarily tested the factory and the grid data services. We queried local and remote MySQL databases.

On the other hand, we tested OGSA-DAI grid services using the GT3 GUI tool described above. The factory data services provide to see the database schema of the database defined in the XML files. This is a powerful tool to discover services for specific data. In addition, we successfully tested the OGSA-DAI services using the GT3 command line clients.

4.2.2.4 Distributed Query Processing

The Grid Distributed Query Service (GDQS) is an example of high-level data integration services. It uses services provided as part of the OGSA-DAI framework, to access potentially heterogeneous data sources. It interacts with OGSA-DAI components at two points:

- During its set up phase, it obtains Service Data Elements (SDEs) from the Grid Data Service Factories (GDSF), which exposes the database schema of the data sources.
- During the query execution, it configures the GQESs (see Section 1) so that they can submit request documents to GDS instances to access data.

The GDQS also combines data analysis with data access and integration, by enabling the client to include calls to web services that perform analysis, within the query [OGSA-DAI-User-ug-gdqs].

4.2.2.4.1 Test

We installed DQP 1.0 on a Linux server and Distributed Query Processing (DQP) client in Linux and windows platforms. We performed the testing using our local DQP installation and the OGSA-DAI DQP test at http://rpc48.cs.man.ac.uk:9090/ogsa/services/ogsadai/DAIServiceGroupRegistry.

4.2.2.4.2 Testing Results

The installation process is complex. DQP configuration requires a significant number of software components.

Although the installation was completed successfully, we found run-time problems during the product testing. Some of the issues could not be resolved during the testing phase.

Like Globus and OGSA-DAI, DQP can be tested with the Java API client or with the GUI tool. We performed most of our test using the GUI. We created DQP instances and queried the services. DQP data services have the database schemas of the data sources.

The query language (OQL) allows performing joins from different data sources. As part of the query statement, you can invoke a web service to process one of the query arguments.

4.2.2.5 SRB

The SDSC Storage Resource Broker (SRB) is a client-server middleware that provides a uniform interface for connecting to heterogeneous data resources over a network and accessing replicated data sets. SRB, in conjunction with the Metadata Catalog (MCAT), provides a way to access data sets and resources based on their attributes rather than their names or physical locations.

4.2.2.5.1 Test

We installed SRB 3.0 and SRB 3.0.1 release of the SRB server with a postgresql based MCAT server. The installation was done on the following environments:

1. SUN Solaris 8.x - Only a partial install was possible. The installation script from SDSC for SRB does not seem to work well in Solaris.
2. Linux - SRB and postgres MCAT server were built and tested. Java based client applications were also tested successfully in this environment. GNU make and GNU gcc compiler are needed.
3. mySRB web application was also deployed in Apache and tested against the SRB server backend.
4. The windows client inQ does not have support for the new 3.x release of SRB at the time of these tests.
5. JARGON (Java API for Real Grids On Networks) was also tested and sample caBIO objects were uploaded as XML, into the SRB server with user definable metadata.
6. We tried to use the GetXML servlet interface provided by caBIO. In our tests, we found out that SRB has difficulty storing complex URLs (URL's with special characters used for GET type form submissions). Using hexadecimal values for these special characters was also unsuccessful.

4.2.2.5.2 Testing Results

We tested the following clients:

- **srbBrowser:** This java based client side tool was tested successfully to browse SRB collections. File objects can be digested and retrieved with this tool.
- **mcatAdmin:** This administration tool was used to create new users and resources, delete users and other resources.
- **SCommands:** Command line tools called SCommands were tested successfully to navigate and manipulate SRB.
- **mySRB:** This web based application was hosted in an Apache server and was tested extensively. MySRB is the most versatile of all the client applications.
- **inQ:** Windows based client has not been updated for the new release. Older versions are error prone and incompatible with the version of SRB that was installed.

Normal Unix file system sharing through the SRB server: SrbBrowser and MySRB front-end tools were used to test normal file system sharing between two discrete clients. This test was successfully executed.

Share caBIO objects:

- caBIO objects were shared through the SRB server in XML format. A backend upload was performed using the caBIO production server instance on the caBIO Gene object in XML format. This Gene object was serialized and uploaded into the SRB server using the JARGON Java API. The Gene XML was downloaded and de-serialized into a caBIO Gene instance on another Java client. This works well except that any marshaling of caBIO objects will be static and might be stale at the time of retrieval and de-serialization. There is a possibility of exposing caBIO as a file system for dynamic/runtime access. By writing a custom driver, we can expose caBIO as a file system to the SRB server.

- We also tested the possibility of using the HTTP interface to caBIO production. A URL was ingested for GetXML servelet. This test failed as the URL was improperly stored in the SRB server. Again, this seems like a bug in the SRB server as it fails with any complex URL, which needs to be encoded.

Remote SQL:

- Using the oracle client, we tried to create a connection to an Oracle database. This test was not fully successful. With the help of the SRB team at SDSC, we tried to troubleshoot this issue. It seemed like a bug in SRB was preventing it to use two databases at the same time.

SRB is a good product for file sharing between what would otherwise be incompatible nodes. File digestion and retrieval is straightforward with backend (OS level) command line tools called SCommands. This is simplified further with the client applications that are available. There is also scope of writing custom clients and file system drivers. Writing custom drivers to enable dynamic ingestion and retrieval of caBIO type complex objects into the SRB server might not be a trivial task. It is estimated that there is a considerable level of effort involved to get this functionality.

4.2.3 P2P

Peer-to-Peer (P2P) are internet systems that connect the resources of a large number of autonomous participants. Inspired by a success of early P2P systems such as Napster or Gnutella, a large and active research community continues to explore the principles, technologies and applications of such implementations [16].

Both P2P and grid technologies are frameworks to share computational resources on the Internet. However, each one fulfills a different set of requirements. P2P is focused on resource sharing of homogeneous desktop systems amongst a large number of users, intermittent connections, low bandwidth and global fault-tolerance. Grid technologies allow the sharing of heterogeneous computational resources for a limited user community.

Although P2P technologies have been in existence for quite some time, implementations of P2P technologies like JXTA, they have not been significantly used in a production environment. The concerns and risks with leveraging JXTA in production environments include accessibility, performance, and security. However, designing a P2P framework specifically to combat these challenges may mitigate these risks. For example, to ensure that services and data are accessible, nodes on the P2P network may be local servers rather than transient desktop clients. In terms of performance, tests can be conducted in the initial proof-of-concept stage to ensure that the framework performs as expected and that any performance bottlenecks are identified early in the process. For security, peer groups may be established with interfaces to access control lists that may be stored in LDAP. As part of this effort, SAIC will explore the risks associated with P2P frameworks and design a robust framework to minimize these risks.

We found that there are a few P2P implementations for collaborative applications in bioinformatics.

4.2.4 Metadata

4.2.4.1 Metadata Catalog

Metadata Catalog (MCAT) semantically describes data sources and data objects in the SRB server. The two types of metadata that can be ingested in the MCAT database and associated with SRB objects are system level and user defined metadata. Both these forms provide a name: value pair of metadata objects. SRB also provides search capabilities based on metadata in the MCAT database.

4.2.4.1.1 Test

See SRB above.

4.2.4.1.2 Test Results

MCAT appears to be a good metadata repository (see SRB analysis above). However, MCAT can only be used along with SRB.

4.2.4.2 MCS

Metadata catalog service (MCS) stores descriptive information (metadata) about logical data items. MCS has been developed as part of the Grid Physics Network (GriPhyN) and NVO projects. The aim of these projects is to support large-scale scientific experiments. MCS is a standalone catalog that stores information about logical data items (for example, files). It also allows users to aggregate the data items into collections. MCS provides system-defined as well as user-defined attributes for logical items and collections. One distinguishing characteristic of MCS is that users can dynamically define and add metadata attributes. MCS can also provide the names of the user-defined attributes. As a result, different MCS instances can be created with alternative contents. MCS have been implemented to run on top of standard web services or on top of the OGSA-DAI grid service. In the latter case, MCS leverages the OGSA-DAI's authentication capabilities to provide secure access to the metadata.

4.2.4.2.1 Test

Documentation about version 2 and very little information about version 3, which is developed on top of OGSA-DAI. At evaluation time, version 3 was not available.

4.2.4.2.2 Test Results

MCS appeared to be an independent metadata repository to use in a collaborative environment. Metadata is provided as a grid service. We recommend developing detailed analysis when version 3 is available.

4.2.4.3 caDSR

The NCI Cancer Data Standards Repository (caDSR) is part of a broad initiative to standardize the common data elements (CDEs) used in cancer research data capture and reporting. It supports data management workflow and adherence to ISO/IEC 11179 metadata standards, the basis for its architecture. Defined by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), the purpose of the 11179 standard is to regularize the terms and representations used in the capture and subsequent annotation of shared electronic data. This type of information constitutes metadata—often characterized as “data about data.”

The sharing of electronic data generally requires that the user community agree upon the types of representation to be used, the manner in which alternate representations can be indicated, the intended meaning of such representations, and the names or “handles” by which such elements can be accessed. The ISO/IEC 11179 standard provides a framework for establishing, maintaining, and disseminating this type of information, and the caDSR is one particular implementation of this standard.

The fundamental information component in the ISO/IEC 11179 model is the data element, which constitutes a single unit of data considered indivisible in the context in which it is used.

Another way of saying this is that a data element is the smallest unit of information that can be exchanged in a transaction between cooperating systems. More specifically, a data element is used to convey the value of a selected property of a defined object, 5 using a particular representation of that value.

A critical notion in the metadata model is that any concept represented by a data element must have an explicit definition that is independent of any particular representation. In order to achieve this in the model, the ISO/IEC 11179 standard specifies the following four components:

1. A DataElementConcept consists of an object and a selected property of that object;
2. The ConceptualDomain is the set of all intended meanings for the possible values of an associated DataElementConcept;
3. The ValueDomain is a set of accepted representations for these meanings; and
4. A DataElement is a combination of a selected DataElementConcept and a ValueDomain.

4.2.4.3.1 Test

We tested the caCORE context from caDSR at NCICB. This context provides the caBIO metadata that includes concepts (classes), relationships and attributes.

We tested Java API and HTTP interfaces to access the caCORE metadata and we integrated caCORE with Globus data services.

4.2.4.3.2 Test Results

caCORE context fulfills advertising and discovery use cases and support part of the query use case realization. The information provided by caDSR such as concepts, relationships and attributes defines the information advertised by a Cancer Research Center. At the same time, caDSR provides the information to discover concepts in the caGrid system. The metadata also provides important information to build the query statement.

We implemented caDSR as a metadata provider in the Globus framework. Therefore, caDSR is the data service provider to define a caBIO grid service.

4.2.4.4 EVS

The National Cancer Institute (NCI) Enterprise Vocabulary Services (EVS) were developed in response to the need for consistent shared vocabularies among the various projects and initiatives at NCI. Controlled vocabularies are important to any application involving electronic data sharing.

The NCI EVS is a set of services and resources that address NCI's needs for controlled vocabulary. The NCI Thesaurus, which is a biomedical thesaurus created specifically to meet the needs of NCI, is produced by the NCI EVS project. The EVS project also produces the NCI Metathesaurus, which is based on the National Library of Medicine's Unified Medical Language System (UMLS) Metathesaurus, supplemented with additional cancer-centric vocabulary. In addition, the EVS project provides NCI with licenses for MedDRA, SNOMED, ICD-O-3, and other proprietary vocabularies.

4.2.4.4.1 Test

We tested EVS using the Java API and HTTP interfaces. The tests developed were looking for technologies that support advertising, discovery and query use cases.

We performed very limited test for tools that support the semantic use cases. However, those preliminary tests demonstrated the big potential of EVS for semantic representation in a collaborative environment. This preliminary test included the OWL representation and Jena2.

4.2.4.4.2 Test Results

EVS support the advertisement and discovery use cases. EVS enrich the metadata provided by caDSR through synonyms. EVS provides synonyms for the concepts exposed by caGRID. This provides the researcher a wider range to get data from caGRID.

During the prototype implementation, we preliminary tested the EVS as a standard ontology, mapping the EVS ids with the Spores data in the caBIO instance.

4.2.5 Object Oriented Data Representation

4.2.5.1 caBIO

The cancer Bioinformatics Infrastructure Objects (caBIO) model and architecture is the primary programmatic interface to caCORE. The heart of caBIO is its domain objects, each of which represents an entity found in biomedical research. These domain objects are related to each other, and examining these relationships can bring to the surface biomedical knowledge that was previously buried in the various primary data sources.

The primary application domains supported by the software involved genomic analysis, and the primary data types were centered on genes, taxon, sequences, diseases, and expression data.

The caBIO domain objects are implemented as Java beans in the *gov.nih.nci.caBIO.bean* package, and include those classes that correspond to biological entities and bioinformatics concepts.

4.2.5.1.1 Test

caBIO is the main data provider for the prototype of caGRID. caBIO is an actor rather than a use case. We selected caBIO because it exposes the data through a standard API without having to contact the data source repository. It is not required to send SQL or XML query statements to query the caBIO data, instead using the caBIO interface that data is available easily.

The test included to query the data using the Java API, the HTTP interface and integrating the caBIO with Globus and OGSA-DAI.

4.2.5.1.2 Test Results

caBIO architecture allowed us to integrate it with Globus and OGSA-DAI. With Globus, we exposed concepts as grid services and with OGSA-DAI we exposed caBIO as data source extending the OGSA-DAI framework. caBIO extension will not affect the integration with OGSA-DAI. caBIO supports the query use case implementation.

4.2.6 Semantic Web

The Semantic Web is an initiative and idea put forth by a large group of individuals and companies as part of the World Wide Web Consortium (W3C). The semantic web page at W3C provides a concise and well-articulated goal of this project:

> *The **Semantic Web** provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming.*

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." -- Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, May 2001

While the goals of the semantic web are admirable and in accord with the goals of the grid, we have found the development, deployment and installation of tools to leverage the metadata structures too immature and experimental to use in the design and implementation of this prototype.

We suggest testing semantic web technology's use, installation and addition into the grid after the prototype has been delivered to NCICB for evaluation.

4.2.6.1 Jena

Jena is an open source framework, which facilitates building semantic web applications. It provides a Java based programmatic API for RDF and OWL. Jena2 is the current version of the Toolkit and it is backward compatible with previous generation Jena1. The Jena2 ontology API is intended to support programmers who are working with ontology data based on RDF. Specifically, this means support for OWL, DAML+OIL and RDFS. Querying of in-memory or persistent RDF models can be done through Jena2 using RDQL (query language for RDF).

4.2.6.1.1 Test

We used a Jena2 toolkit and MySQL database on a Linux workstation.

4.2.6.1.2 Test Results

We performed preliminary tests with Gene Ontology (GO) database in OWL format. We received mixed test results. On one hand, we found that the Jena2 toolkit has a well-defined API for manipulating OWL and RDF documents including persistent storage of these models in various relational databases. Querying of these models was straightforward. However, we also found some discrepancy in the databases supported and the drivers available were not 100% compatible with the toolkit. In addition, Jena creates huge memory intensive XML models that can drain (and in some instances freeze) system resources when dealing with bigger documents.

4.3 Evaluation Criteria

We built the evaluation criteria using the use cases and supplementary requirements. The complete checklist of supplementary requirements is in Appendix A and the detailed use cases are in Appendix B.

4.4 Evaluation Results

We used Globus toolkit version 3.02 as a framework for the caGRID project prototype. Globus toolkit fulfills most of the requirements outlined above. Globus core along with Globus information service components provides tools to implement advertisement, discovery and query use cases. Using the information services component we can aggregate services and advertise them to the public in a standard way.

- **caBIO Java API**: Use Java API as the integration mechanism to expose caBIO services.
- **Metadata**: Describe caBIO grid services (caBIO services) with data services. Globus toolkit provides an excellent mechanism to create and associate service metadata. We can describe the grid services including attributes that will be stored as part of the service instance. One example of a grid service metadata element could be "Research center provider". In addition, we can enrich the service definition by exposing metadata from EVS, caDSR, or Jena (using NCI thesaurus owl).

- **Notification**: Usually the data advertised can change. Using Globus toolkit we can implement registration/notification services. In this case, if the metadata that describes a grid service change, the system will notify the changes to the other participants in the grid.
- **caBIO as OGSA-DAI data source**: OGSA-DAI was used as the caGRID framework. We extended the OGSA-DAI framework to provide caBIO as a new data source. As described before, we used caDSR and EVS as metadata providers. Implemented the metadata with data service feature from Globus toolkit.

The following table summarizes the technology evaluation process.

	Advertise	**Discovery**	**Query**	**Object to Data Mapping**	**Semantic mapping**
Web services	Tools like UDDI extend to advertise service. Centralized registry.	UDDI for service and discovery. WDSL for service access.			
MobyS / S-Moby	Provide tools to advertise services using semantic web.	Provide tools to discover services using semantic web.			
Globus, OGSA-DAI, DQP	Globus provides tools for advertisement. OGSA_DAI has a data service registry ready to use.	Globus MDS and data services.	Object model / caBIO Java API. Data bases (RDB, XML).		
SRB / MCAT	MCAP / Metadata service.	MCAT / Metadata service.	SRB server / FS, DB, Obj.		
Jena2 / RDF / OWL	Metadata representation.	RDF and OWL query.			Ontology representation and mapping.
MCS	Metadata provider as Grid services.	Metadata service			
caDSR	Metadata provider for concepts, attributes and relationships.	Service discovery by concepts, relationships and attributes.	Provide information to build queries to service providers.	Provide object-oriented representation to create the mapping.	
EVS	Provides synonyms and definitions of concepts.	Enrich discovery process allowing user to search also for synonyms.			Maps concepts and attributes with standard definitions. Ontology mapping.
Object oriented representations - caBIO			Three interfaces: Java API, HTTP and Web Services.	Customize XML representation to model other DB.	

5 caGRID Prototype

The caGRID Prototype is an extension to the default functionality of the OGSA-DAI framework. The OGSA-DAI framework is built upon the Globus Toolkit. This chapter first gives an overview of the overall architecture of the caGRID Prototype, and then gives an architectural overview of each of its software dependencies. It is beyond the scope of this document to give a detailed technical description of the OGSA-DAI framework and the Globus Toolkit upon which caGRID depends. Listed in Chapter 9 are references for further reading about these systems.

The remainder of this chapter describes the main implementation and deployment of the requirements and use cases, which constituted the main part of the caGRID teams' development effort. Appendix C provides a graphical view of the prototype test environment and screenshots from a demo of the caGRID prototype.

5.1 caGRID SYSTEM architecture

The caGRID architecture includes Globus 3.0 and OGSA-DAI 3.1 plus extensions. In Figure 5.1-1, the sections highlighted in red and light blue represent the extensions to the system implemented by the caGRID team.

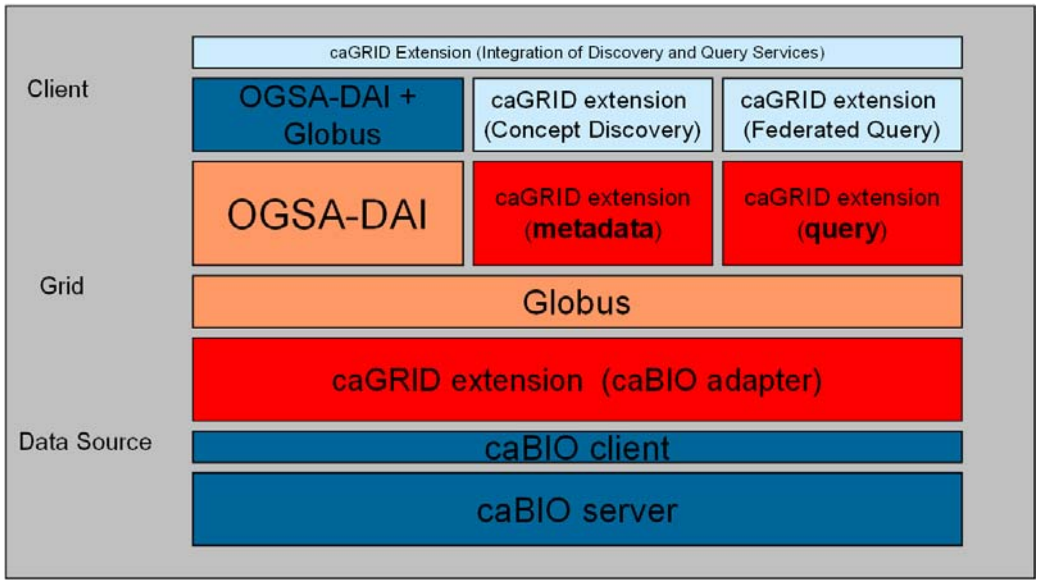

caGRID Extension (Integration of Discovery and Query Services)
Client
OGSA-DAI + Globus
caGRID extension (Concept Discovery)
caGRID extension (Federated Query)
OGSA-DAI
caGRID extension (metadata)
caGRID extension (query)
Grid
Globus
caGRID extension (caBIO adapter)
Data Source
caBIO client
caBIO server

Figure 5.1-1: caGRID System Architecture

5.2 caGRID Components

5.2.1 Globus Toolkit 3.0

The Globus Toolkit (GT3) is a prototype reference implementation of the Open Grid Service Infrastructure. GT3 Core offers a run-time environment capable of hosting grid services. The run-time environment mediates between the application, the underlying network, and the transport protocol engines. GT3 Core also provides development support including programming models for exposing, and accessing grid service implementations.

Grid services can be viewed as standard web services adapted to the requirements typically found in a grid environment. In order to make this as transparent as possible to grid application developers, GT3 Core implements most of the OGSI functionality on behalf of users and providers of grid services [22].

Grid services can be a static set of persistent services as well as transient services such as a query against a database, a data mining operation, a network bandwidth allocation, a running data transfer, and an advance reservation for processing capability. There may be one or more instances of a particular grid service. Each grid service instance has a set of service data associated with it. This data has a standardized representation as Service Data Elements (SDEs).

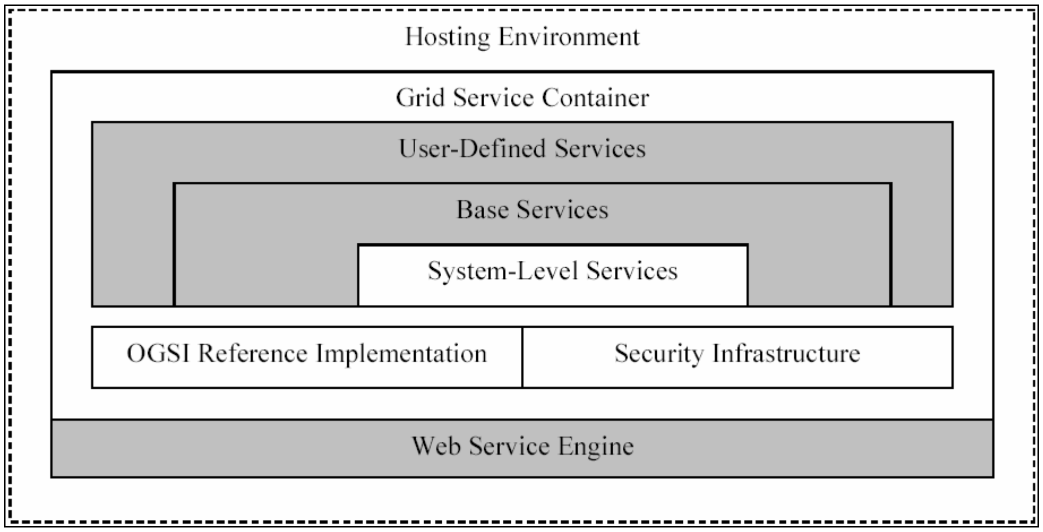


Figure 5.2.1-1: Globus Toolkit 3 Core Architecture

The white boxes in Figure 5.2.1-1 represent components provided by GT3 Core. Together they make up what we consider the essential building blocks for grid services. The *OGSI Reference Implementation* provides implementations for all OGSI specified interfaces, as well as APIs and tools to make it easier to develop OGSI compliant services. The *Security Infrastructure* implementation provides SOAP as well as transport level message protection, end-to-end mutual authentication, and single sign-on service authorization; essentially a rendering of the GSI implementation known from Globus Toolkit 2 in an OGSI environment. These two building blocks do not provide any run time services but serve purely as a base for other services. GT3 Core, however, also contains some infrastructure level run-time services that are generic enough to be used by, and in conjunction with all other grid services. These so-called *System-Level Services* are built on top of the OGSI Reference Implementation as well as the Grid Security Infrastructure. GT3 also ships a number of higher-level or *Base Services* like Program Execution, Data Management, and Information Services. A detailed discussion of these services is outside of the scope of this document, since they are not part of the generic core infrastructure. Base Services are typically built on top of the OGSI and GSI components, as well as in some instances the System-Level Services component. *User-Defined Services* is the term for higher-level services, not included in the toolkit, that are built on top of any subset of GT3 components including Base Services.

All these services and primitives interact with an abstract OGSI run-time environment we call the *Grid Service Containe*r. The purpose of the container is to shield the application from environment specific run-time settings, such as what database is used to persist service data. The container also controls the lifecycle of services, and the dispatching of remote requests to service instances. An important design goal of the container has been to make it as implementation agnostic as possible and thus allow for many different implementations. We also hope that the interfaces that constitute the container will be standardized in order to make it possible to host services in alternative implementations, similar to how Enterprise JavaBeans (EJBs) can be hosted in alternative J2EE implementations today. The container extends, as well as encapsulates the interfaces defined by a standard *Web Service Engin*e, which is responsible for implementing XML Messaging.

Finally, the Web Service Engine and Grid Service Container are hosted in a *Hosting Environmen*t, which implements traditional Web Server functionality such as the transport protocol (for example, HTTP).

The Globus Toolkit provides the following core services [22]:

- **Security**: The Globus Toolkit uses the Grid Security Infrastructure (GSI) for enabling secure authentication and communication over an open network. GSI provides a number of useful services for Grids, including mutual authentication and single sign-on.
- **Data Management**: This component includes GridFTP, Reliable File Transfer protocol (RTF) and Replica Location service (RLS). GridFTP is a high performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. RTF protocol provides an interface for controlling and monitoring third party file transfers using GridFTP servers. RLS maintains and provides access to mapping information from logical names for data items to target items. These target names may represent physical locations of data items, or an entry in the RLS may map to another level of logical naming for the data item.
- **Resource Management**: Globus Resource allocation manager (GRAM) simplifies the use of remote systems by providing a single standard interface for requesting and using remote system resources for the execution of "jobs."
- **Information services**: Provides the functionality within which Service Data Elements can be collected, aggregated, and queried, data feeds can be monitored, and Service Data Elements can be created dynamically on demand.
- **XIO**: Globus XIO is an extensible input/output library for the Globus Toolkit that provides a simple and intuitive API (open/close/read/write) to swappable IO implementations. Globus XIO has two main goals: Provide a single user API to all grid IO protocols and minimize the development time for creating/prototyping new protocols.

5.2.2 OGSA-DAI

The goal of OGSA-DAI (Open Grid Services Architecture Data Access and Integration) is to provide uniform service interfaces for data access and integration through the grid. Disparate, heterogeneous data sources and resources can be treated as a single logical resource using OGSA-DAI interfaces. OGSA-DAI grid services themselves will then provide the basic operations that can be used to perform sophisticated operations such as data federation and distributed queries within a virtual organization (VO), hiding concerns such as database driver technology, data formatting techniques and delivery mechanisms from clients.

This goal has to be achieved by the provision of an efficient grid-enabled middleware reference implementation of the components required to access and control data sources and resources.

OGSA-DAI is intended to contribute to a future in which it is envisaged that scientists will move away from technical issues such as handling data location, data structure, and data transfer and integration, and instead focus on application-specific data analysis and processing [23].

OGSA-DAI is considered logically as a number of co-operating grid services. These grid services act as proxies for the systems that actually host the data. The functionality of the 3.1 release has been tested with the MySQL 3.23 / 4.0, DB/2 and Oracle 9i databases and is consistent with the SQL-92 standard. It has also been tested with the Xindice 1.0 database and supports XPath queries. The release supports regular expression searches of XML documents. Database management functionality is dependant on the underlying database supporting management through an API such as JDBC. In particular, Oracle does not support the creation or deletion of databases via OGSA-DAI [24].

Figure 5.2.2-1 shows the three main OGSA-DAI service components: the Service Group Registry (DAISGR), the Grid Data Service Factory (GDSF) and the Grid Data Service (GDS).

- **Service Group Registry (DAISGR)**: Provides a directory facility for OGSA-DAI service factories. By querying a DAISGR, clients can discover OGSA-DAI services that offer particular services or capabilities or manage particular data sources.
- **Grid Data Service Factory (GDSF)**: Is a specialized factory service that exposes a data resource and can create new GDS through which clients can interact with this data resource. In addition, clients can query a

GDSF to extract information (metadata) about the data resource exposed by the GDSF and determine whether the data resource meets their application-specific requirements.

- **Grid Data Service (GDS)**: Is the primary OGSA-DAI service. GDSs provide access to data resources using a document-oriented model: a client submits data retrieval or update requests in the form of an XML document; the GDS executes the request and returns an XML document holding the results of the request.

A client will interact with the system by connecting to a Service Group Registry and retrieving a list of available Grid Data Service Factories. Having chosen the appropriate factory, the client then requests that the factory create a GDS instance to allow access to a specified data resource. Now the GDS is ready to accept perform documents to run database queries, transform results, and deliver data.

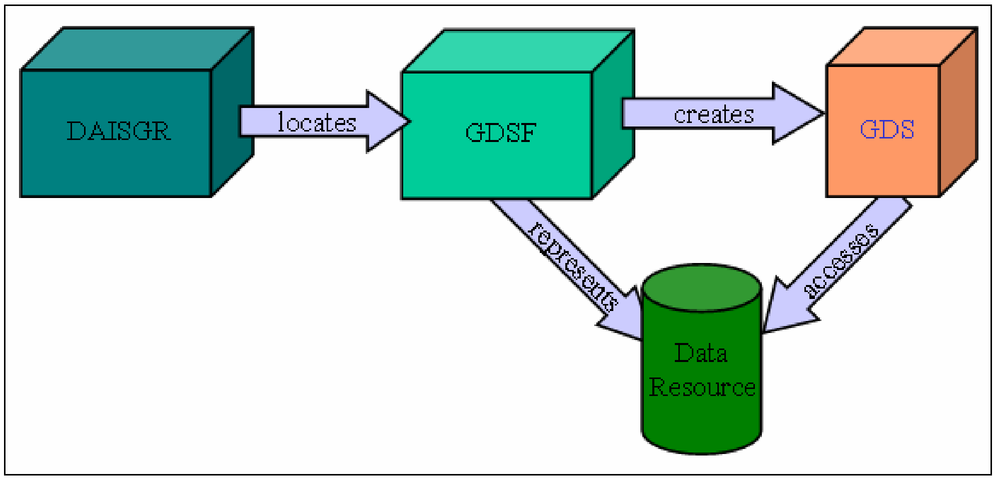


Figure 5.2.2-1: OGSA-DAI Service Components

At the core of OGSA-DAI is the Engine that is shown in Figure 5.2.2-2. The engine orchestrates everything that happens inside OGSA-DAI when a perform document is submitted to a GDS. When a GDS receives a perform document, it passes the document to the engine along with some more contextual information (such as the DN from a user certificate). The engine then processes the perform document by coordinating the following actions:

- The perform document is parsed and validated.
- The activities specified in the perform document are identified.
- The activity implementations are instantiated.
- The pipes between the activities are instantiated.
- The activity handlers are instantiated and started.
- The handlers passing the data produced through the pipes process the activities.
- The outputs are combined to form a response document.
- The response document is returned to the GDS, to be returned to the client.
- The status of the activities is available as OGSI service data at all times during the execution of the perform document.

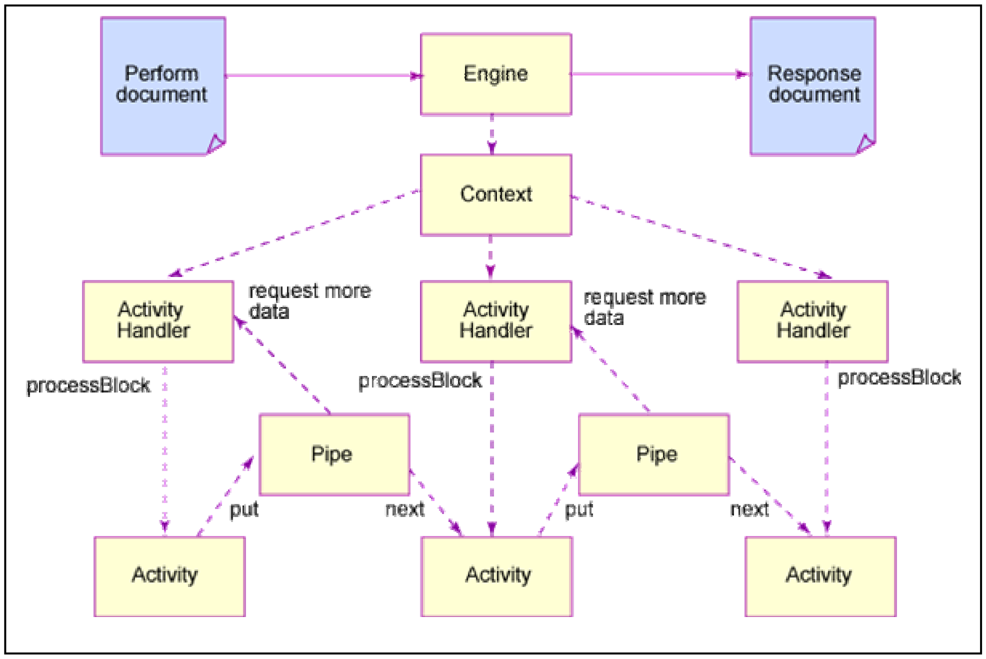


Figure 5.2.2-2: OGSA-DAI Engine

The engine is designed to deal with one perform document at a time. If a perform document is submitted while one is still executing, a User Exception will be returned. Of course, this does not stop you from creating multiple GDS instances from the same factory -- each GDS instance will have its own engine so you can then execute multiple perform documents at the same time.

5.2.3 OGSA-DAI Extensions

The caGRID project extends OGSA-DAI by adding a new data source to the OGSA-DAI framework. By default, the OGSA-DAI framework provides data access to relational database management systems (RDBMSs) (Oracle, mySQL, and so forth), XML databases and files (CVS). The prototype included caBIO as its data source.

Current caBIO version 2.1 allows the installation of a caBIO server to instantiate a RDBMS and query caBIO objects through a remote client using RMI. This architecture allows access to only one caBIO instance. Extending the OGSA-

DAI functionality to enable caBIO as a grid data source allows multiple caBIO queries to be submitted to multiple remote caBIO instances. Below is a list of the main OGSA-DAI components that were extended to implement caBIO as grid data source:

- **Data Sources**: caBIO added as a new data source to the OGSA-DAI framework.
- **Activities**: To implement the caBIO Java API, two activities were designed. The first activity, *caBIOSimpleQueryActivity*, gives access to the caBIO HTTP interface. This activity allows the web service to be queried using the caBIO HTTP meta language embedded within a perform document. The second activity, *caBIOQueryActivity*, implements the caBIO Java API. This activity allows caBIO objects to be described in an XML perform document query and allows more sophisticated queries to be defined by having full access to the caBIO object model.
- **caBIO Metadata - Local caDSR**: A local XML representation of caDSR was defined in OGSA-DAI to represent the objects advertised from a Cancer Research Center. Each research center has a local XML file with the caDSR schema. The file can be updated any time since it is using dynamic metadata process.
- **Query language**: Prototype query language with XML representation to query a caBIO data source.
- **Service Metadata**: Two new metadata categories were added (*caBIOSchema* and *ResearchCenterInformation*) to describe the caBIO data service factories. This includes the caDSR/EVS metadata extractor.
- **Concept Discovery**: Client extended to discover by concept name in the grid. The result set is the list of factories grouped by concept relationships.
- **Federated query**: Federated query by relationship (from discovery process). Send serialized queries to the factories that have advertised concept-relationship.
- **Result set cached (Alpha)**: The result set from a federated query will be stored in a client XML database. Using another factory, the result set can be queried using XPath expressions.
- **Metadata query (Alpha)**: Ability to query metadata without using XPath expression.
- **Data Access Clients**: A variety of data access clients were implemented. A number of simple command line clients were implemented to query the grid for caBIO data sources. These were extended to perform service discovery and federated queries. A fully featured Swing Client was also implemented allowing OGSA-DAI grid services to be browsed graphically and gave full access to all of the OGSA-DAI extensions implemented as part of the caGRID Project.

5.2.4 Metadata

In a “grid enabled” data sharing facility, datasets may not be well known amongst all participants of the grid. Therefore, it becomes critically important to describe the context that the data was captured. In a grid-enabled environment, we describe this contextualization of the data as “metadata” (data about data).

Life sciences are increasingly becoming a distributed global effort with large computational and data requirements. Biological data is often very complex, stored in a distributed global environment, often incomplete and changing frequently.

To integrate the highly fragmented and isolated data sources in the life sciences community, we need semantics to answer higher-level questions. With richer semantics, we can extend the grid infrastructure that has been traditionally done with greater computational facilities. This is shown in Figure 5.2.4-1

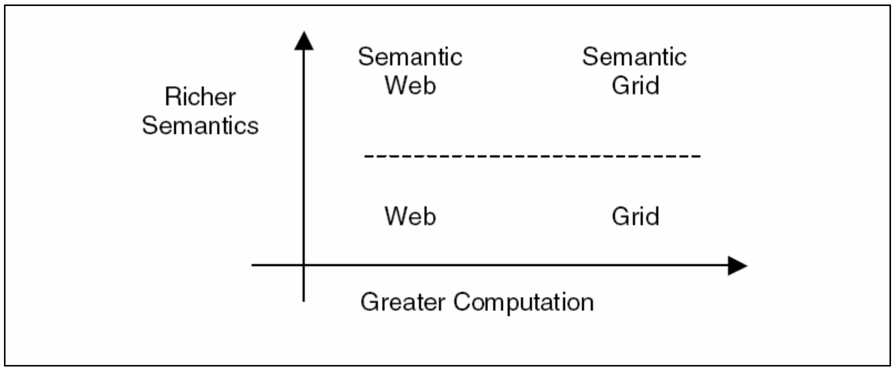


Figure 5.2.4-1: The Semantic Grid [9]

For an interoperable infrastructure, we must apply Semantic Web technologies inside the grid middleware. The grid community has adopted the service-based approach known as the Open Grid Service Architecture (OGSA). OGSA makes it easy to create dynamic "virtual organizations" by service integration.

Associating appropriate metadata to the grid services has been vitally important to the caGRID prototype. This is realized by adding contextual metadata around Service Data Elements (SDEs) allowing the realization of the discovery use case.

5.2.4.1 caDSR and EVS as Metadata Providers

One of the core requirements of Semantics in a distributed system is the need for common language and a common understanding of what each term means. This is particularly useful in a situation where shared ontologies provide the term. This requirement is being fulfilled in the caCORE prototype system by the NCI Enterprise Vocabulary Service (EVS).

The NCI EVS is set of services and resources that address NCI's needs for controlled vocabulary [11]. One of the many reasons EVS was chosen for the caGRID prototype is the ability to access the EVS servers with the caCORE API.

Based on the ISO/IEC 11179 model, caDSR system provides common data elements (CDEs) that can be used as metadata descriptors for the caCORE "objects" being shared on the caGRID prototype. In addition to CDEs, caDSR provides us with controlled vocabularies and logical relationships of entities across domains. Any UML model can be loaded into the caDSR system, which is of great significance in a grid environment where many different datasets will be shared.

5.2.4.2 Local vs. Global Metadata Repositories

The caDSR instance at NCICB represents the global metadata repository. This metadata instance exposes the whole caBIO schema and is a source element to build local metadata instances.

Local metadata repositories are XML representation of the Global caDSR. This is a subset of the global caDSR, which defines the objects exposed by a research center. Listed in the following sections are detailed descriptions of these two metadata implementations.

5.2.4.3 Dynamic and Static Metadata

5.2.4.3.1 Static Metadata

Site wide metadata is data that remains static throughout the lifetime of the site that is sharing a particular dataset. Examples of this data may be lab groups that are publishing the data, data types that are being exposed, and so forth.

Some of the SDEs (Service Data Elements) identified for site-wide metadata are:

- **Contact Info**: Contact info contains the name, address, telephone and other administrative data about a site who shared information on the grid. This will also contain URL information for the particular site that is sharing data on the grid. For example, the URL http://nci.nih.gov would be published as site wide meta data.
- **Data Type**: This is the data type that this site is exposing. This attribute will be constrained to the following concepts (and may be extended at a later date):
- Genomic
- Transcript
- Proteomic
- Metabolomic
- Clinical (Phenotype)
- Disease

This will allow researchers who take part in the grid to perform a very quick search for data sets that fall into one of the categories above. It should be mentioned that these categories aim to capture 90% of the activities that are taking place in the life sciences.

5.2.4.3.2 Dynamic "Data source" Metadata

Data source metadata is data that is associated with a particular data set of piece of a data set. This data is exposed to the user upon connection to the site of interest and is not considered "site wide" (since it may change more often due to additional experimentation linkage to more data sources).

A preliminary list of dynamic metadata for SDEs is as follows:

- **Data Collection Method:** Data collection method describes the platform that was used to capture a particular piece of data. This would include the name of the technique that was used to perform the analysis if the data was discovered "in-silico" or the platform (affymetrix) if identified through wet lab techniques.

- **Contributing Data Source(s):** At times, a particular piece of data is an aggregation of many pieces of information. This field allows the users of the grid to know what other data sources contributed to the final resulting data that is published to the grid. Examples of this include Genbank, PIR, NREF, PSD, SwissProt, UniGene, and so forth.

- **Ontological Category:** An ontological category describes a particular concept that the dataset maps. This includes GO terms that are familiar to the biological community and MESH terms that describe clinical concepts. The most helpful data source to use for this field is Go-A, MESH, and UMLS.

- **Sample Number:** This field describes the number of samples that contributed to the data source or piece of data that is queried in a system that is shared on the grid.

5.3 caGRID Functions

This section describes the three main functions implemented in the caGRID prototype: advertisement, discovery and query.

5.3.1 Advertisement

Advertisement represents the action of publishing a data source to the caGRID system. For the specific implementation of caBIO advertisement, a Cancer Research Center advertises concepts, relationships and attributes from the caBIO model to the Cancer Research Community. Concepts in caGRID represent caBIO classes.

The concepts advertised are defined in a configuration file (*mapping.xml*); this is an extension from the OGSA-DAI framework. The mapping file represents a local caDSR instance of the services advertised. Therefore, the local mapping file is a subset of the caDSR from NCICB.

By default, caGRID advertises one caBIO factory as a data source in the local registry server. A caBIO factory can be advertised to a central caBIO registry or any other registry by using Globus/OGSA-DAI tools. Optionally, you also can define a lifetime of a factory when you advertise caBIO to a registry server.

The local caBIO factory, which is a grid service, has associated data services or metadata. The factory has several metadata categories, some categories are from the Globus Toolkit, while others are from the OGSA-DAI framework and others are caGRID extensions. The mapping file mentioned above is a metadata service called *caBIOSchema*; therefore, *caBIOSchema* defines the services that a given Research Center advertises to the grid.

It is important to point out that Cancer Research Centers that advertise caBIO concepts must have a local caBIO installation. This local installation can potentially map to other local data sources by using the caBIO Object Relational Bridge (OJB) implementation.

It is possible to have more than one caBIO grid data-service factory at a Cancer Research Center exposing the same or different caBIO instances. In this case, each factory can expose different caBIO concepts and can be advertised to different registry services.

Once the factory is available, advertised concepts can be modified at any time by updating the caGRID mapping file. caBIO concept metadata may be advertised using two interfaces. The first accesses *caBIOSchema* from the NCICB caBIO factory, which exposes the whole caBIO model. The second uses the caGRID client (OGSA-DAI extension) which returns an XML representation of the caDSR instance of the caBIO schema. Consequently, the advertised process includes concepts, relationships and attributes into the mapping file.

Following is an example of a caDSR instance that is exposing disease concept:

```
<?xml version="1.0" encoding="UTF-8"?>
<context>
    <gov.nih.nci.caDSR.bean.DataElementConcept id="CD82C5BE-10C9-349D-E034-0003BA12F5E7">
        <preferredName>Target</preferredName>
        <preferreDefinit_unit>A gene thought to be at the root of a disease etiology,
            _           and which is targeted for therapeutic intervention in a
            _           clinical trial.
        </preferreDefinit_unit>
        <longName>gov.nih.nci.caBIO.bean.Target</longName>
        <version>2.0</version>
        <dateModified>Fri May 28_00:00:00 EDT 2004</dateModified>
        <dateCreated>Tue Dec 02_00:00:00 EST 2003</dateCreated>
        <publicId>2178542</publicId>
        <evs-synonym>TGT</evs-synonym>
        <evs-synonym>Target</evs-synonym>
        <gov.nih.nci.caDSR.bean.DataElement id="CD82C5BE-15DE-349D-E034-0003BA12F5E7">
            <preferredName>TargetId</preferredName>
            <dataType>The NCI's intçrnal identification for the target.</dataType>
            <publicId>2178733</publicId>
        </gov.nih.nci.caDSR.bean.DataElement>
        <gov.nih.nci.caDSR.bean.DataElement id="CD82C5BE-15E5-349D-E034-0003BA12F5E7">
            <preferredName>TargetTargetName</preferredName>
            <dataType>The name of the target.</dataType>
            <publicId>2178734</publicId>
        </gov.nih.nci.caDSR.bean.DataElement>
        <gov.nih.nci.caDSR.bean.DataElement id="CD82C5BE-15EC-349D-E034-0003BA12F5E7">
            <preferredName>TargetTargetType</preferredName>
            <dataType>The type of the target.</dataType>
            <publicId>2178735</publicId>
        </gov.nih.nci.caDSR.bean.DataElement>
        <gov.nih.caDSR.bean.DataElementConceptRelationship id="CD82C5BE-166F-349D-E034-0003BA12F5E7">
            <gov.nih.nci.caDSR.bean.DataElementConcept id="CD82C5BE-10BD-349D-E034-0003BA12F5E7">
                <preferredName>Agent</preferredName>
                <longName>gov.nih.nci.caBIO.bean.Agent</longName>
                <preferreDefinit_unit>A therapeutic agert (drug, intervention therapy) - used in  a clinical trial protocol.
                </preferreDefinit_unit>
                <publicId>2178540</publicId>
            </gov.nih.nci.caDSR.bean.DataElementConcept>
        </gov.nih.caDSR.beaz.DataElementConceptRelationship>
        </gov.nih.nci.caDSR.bean.DataElementConcept>
</context>
```

5.3.2 Discovery

As mentioned earlier, caBIO factories in caGRID system have several metadata categories such as Globus, OGSA-DAI, and caGRID extension. The discovery process can be performed by sending XPath queries to the factories asking for specific data. Out of the box, Globus/OGSA-DAI discovery allows you to query the metadata with an XPath expression. The four main metadata categories are:

- **ProductInformation** (OGSA-DAI)
- **localCaBioSchema** (caGRID)
- **CaBioSchema** (caGRID)
- **ResearchCenterInformation** (caGRID)

Using the Globus toolkit, you can get data from a given category as an XML representation. To get specific data, you can use XPath expressions. The Globus client allows submitting a discovery process describing the factory, the data category, and the XPath expressions along with the name spaces. The query is submitted to a specific factory. That functionality was extended by wrapping the XPath expression and sending the query to all services registered in a given DAI Service Group Registry (DAISGR). Another extension is specifically using the caBIOSchema metadata from a caBIO factory. This process allows identification of a Cancer Research Center that is advertising the desired caBIO concepts. The screening process is based on the metadata (*mapping.xml*) defined in each local site.

Two main extensions have been added to discover caGRID services. The first is to query for research centers that provide a specific concept. The second is to discover a metadata category without having to write an XPath query. Below is an example of a results report from a discovery query searching for “ClinicalTrialProtocol:”

```
<searchResult>
  <search name="ClinicalTrialProtocol" id="DA512A97-27D5-16D5-E034-0003BA0B1A09">
   <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-Disease" id="DA512A97-57B2-16D5-E034-0003BA0B1A09">
   <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-Agent" id="DA512A97-57BA-16D5-E034-0003BA0B1A09">
   <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-ProtocolAssociation" id="DA512A97-57A9-16D5-E034-0003BA0B1A09">
   <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol" id="CD82C5BE-10E7-349D-E034-0003BA12F5E7">
    <handle url="http://156.40.48.168:8080/ogsa/services/ogsadai/GeorgeTown_caBIODataServiceFactory" />
    <handle url="http://jforge.net:8080/ogsa/services/ogsadai/Panther1_caBIODataServiceFactory" />
    <handle url="http://cbiogrid101.nci.nih.gov:8080/ogsa/services/ogsadai/Duke_caBIODataServiceFactory" />
    <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-PersistentCaBIOBean" id="DA512A97-57C2-16D5-E034-0003BA0B1A09">
   <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-Agent" id="CD82C5BE-1682-349D-E034-0003BA12F5E7">
   <handle url="http://cbiogrid101.nci.nih.gov:8080/ogsa/services/ogsadai/Duke_caBIODataServiceFactory" />
   <handle url="http://jforge.net:8080/ogsa/services/ogsadai/Panther1_caBIODataServiceFactory" />
   <handle url="http://156.40.48.168:8080/ogsa/services/ogsadai/GeorgeTown_caBIODataServiceFactory" />
   <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-ProtocolAssociation" id="CD82C5BE-1683-349D-E034-0003BA12F5E7">
   <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-Agent" id="CD82C5BE-1684-349D-E034-0003BA12F5E7">
   <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
</searchResult>
```

All caBIO queries are based upon object relationships. To issue a query to a caBIO instance, regardless of the interface (Java API, HTTP, web services), you first specify the object where you want the result set and then the search criteria, which represent the filters based on object properties or other object properties that have relationships with the original object. Therefore, the concept discovery client and the *caGridBrowser* swing client provide the information to issue the same query to different Cancer Research Centers.

5.3.3 Query

The query process allows the retrieval of data from one or more caBIG instances or Cancer Research Centers. The sites that receive the query are either defined by a Discovery process to all caBIG instances or to specific site(s) that were previously identified as data providers. As described above, the OGSA-DAI framework has been extended to share caBIO instances exposing caBIO as an OGSA-DAI data service. Specifically, we have created two activities to query caBIO instances. Regardless of the activity, you can query the system by creating a perform document or by contacting the activity directly. Both cases are OGSA-DAI standards. The perform document allows us to specify the caBIO query concept and the filter of the query. Along with the query activity, you can use any predefined transformation or delivery activity defined in the OGSA-DAI framework. These two activities allow you to transform the query results using standard XSLT and/or deliver the result set to other sites different from the query originator. You can also perform a RDBMS bulk query or create local files with the result set.

```
<gridDataServicePerform
  xmlns="http://ogsadai.org.uk/namespaces/2003/07/gds/types"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ogsadai.org.uk/namespaces/2003/07/gds/types
  ../../../../schema/ogsadai/xsd/activities/activities.xsd">
  <documentation>
    This example demonstrates how to parameterise an caBIO
  </documentation>

  <caBioSimpleQuery name="caBIO2.0">
    <criteria position="1" class="disease" version="1.0" id="abcd" attribute="name" value="neoplasia"/>
    <query class="ClinicalTrialProtocol" version="1.0" id="">ClinicalTrialProtocol</query>
    <output name="Test"/>
  </caBioSimpleQuery>

</gridDataServicePerform>
```

The caBIO result set is an XML representation generated from the caBIO Java API. You can query caGRID and receive the results as one XML document in the standard output, or you can redirect the output to an XML database. The last option allows you to query the result set using another grid service that queries the XML data using an XPath expression.

As stated at the beginning of this paper, the prototype scope is to develop a grid middleware to access various caBIO instances through the caGRID framework. As an example, several command line clients were implemented that integrate the discovery and query services as proof of concepts of the middleware implementation. In addition, a graphical swing client (caGridBrowser) was developed to allow the caGRID prototype to be browsed graphically. This client also has full discovery and query services and allows a query to be sent asynchronously to multiple data service sites.

5.4 caGRID Deployments

At system startup (time t -> 0), there will be just one node in the grid. This “mother” node provides the following:

- OGSA-DAI service registry
- Ability to query NCICB’s cancer data
- Gateway to caDSR metadata repository and EVS

Subsequent members (nodes) to this grid can chose to register their service factories in the NCICB service registry or host their own registries. Having many registries facilitate the "virtual organization" concept discussed earlier in this paper.

A grid node is a participating entity in the grid that either provides data services to the grid or uses the existing services in the grid. Building on this concept, a grid node can be one of these types of installation:

- **Query client:** The primary user of this type of installation will be a researcher querying the various data services located in the caGRID deployment. This client requires minimal installation and consists of GUI and command line tools.

- **Data sharing client:** These clients will be researchers or research centers that have data to share. Such an installation will implement a full set of use cases (Appendix B) that will help link a data source to the grid, using an appropriate Object model. Once appropriate factories have been created to share this data, there will be sufficient flexibility in how these service factories will be advertised. A factory can be registered in one or many registries.

For the prototype, a minimal test grid has been set up. The caGRID prototype deployment consists of four servers, three hosted by NCICB in Rockville, MD and one by Panther Informatics in Boston, MA. Each of these servers represents a grid node. To play out the demo scenario, the servers in NCICB and Boston each masquerade as a Cancer Research Center with different Clinical Trial Data served using the caCORE API.

By making slight modifications to the Object Relational Bridge (OJB) layer, we are able to serve different datasets using the same object model.

Shown in Figure 5.4-1 is the caGRID deployment diagram.

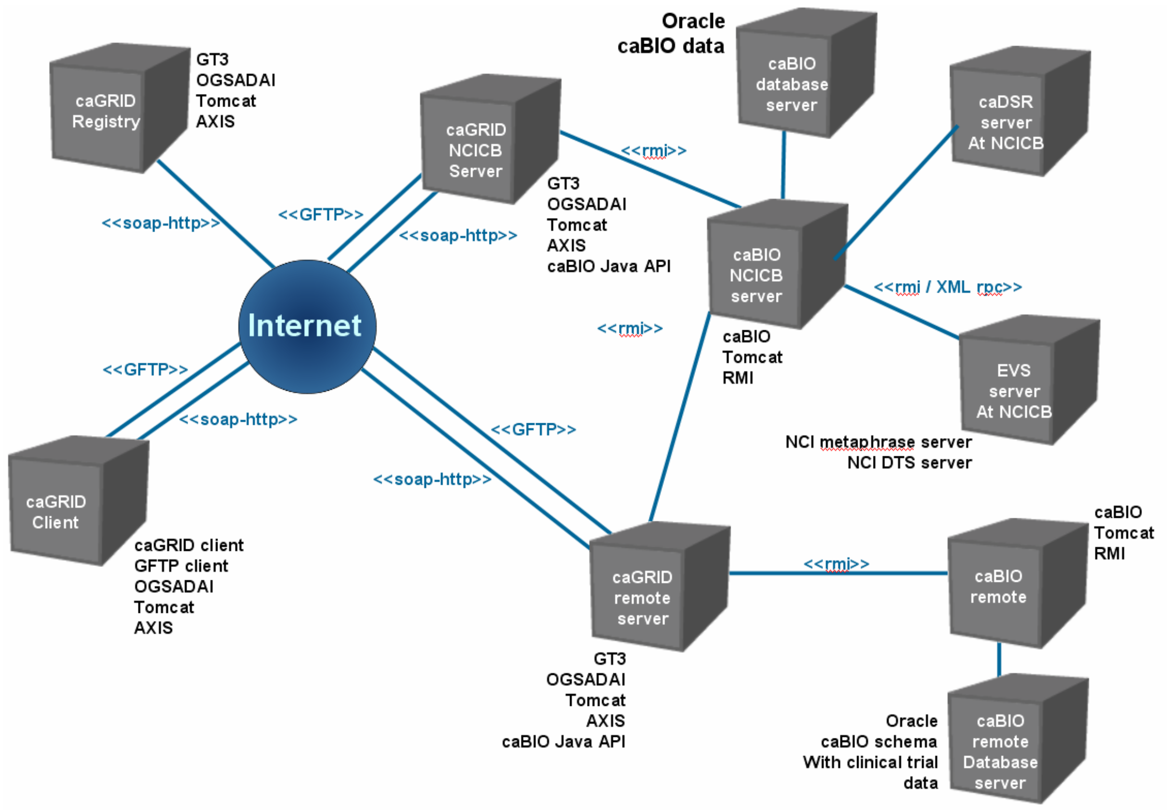


Figure 5.4-1: caGRID Deployment Diagram

6 Lessons Learned

After the prototype implementation was completed, the caGRID team performed a postmortem to discuss lessons learned in using grid technology. The team came to the following conclusions:

1. **There is an inherent learning curve in implementing grid technologies.** Even though grid technologies are built upon familiar web and web services based architecture, there is a lot of jargon to learn. The architecture of a grid framework, including the Open Grid Services Architecture (OGSA), Globus Toolkit, and Global Grid Forum (GGF) standards, make grid technology difficult to penetrate during the initial learning period. However, once the components were understood, the team found that the client API and OGSA-DAI runtime system were rich enough to allow the team to concentrate on domain level problems rather than grid details. In summary, the learning time invested gave back a high return in the implementation.
2. **Grid technology is still maturing, new standards are evolving and there is no consensus yet in the IT industry.** Grid technology is truly a moving target. Several groups, both commercial and academic, are releasing new specifications to produce a grid-enabled system. These include OGSA, WS-RF, WS-I, WS-Notification and WS-Events. The Globus group is moving their entire infrastructure toward the new WS-RF standard. In response to this information, the OGSA-DAI group has decided to produce the next release of their toolkit based on the current WS-I specification in late 2004, then also on top of the new Globus toolkit version 4.0, which will be based upon the new WS-RF standard, in 2005. In spite of the fact that the technology is evolving rapidly, we did successfully leverage the functionality provided by OGSA-DAI and Globus. Had we attempted to implement the same functionality with plain web services, P2P, or any other collaboration technology, the implementation would have taken much longer and we would not likely have achieved the same level of functionality.
3. **Common metadata structure and terminology is necessary to effectively describe services and data.** The caGRID team realized early in the development of the prototype that metadata around a particular service would be essential to the production of interoperable services. The team strongly believes that more semantic information in the form of metadata needs to be passed between interoperating sites to assure that each piece of data is described in enough detail to produce virtual research organizations. The caGRID team scratched the surface of this problem with the introduction of caDSR and EVS-derived metadata in the discovery process. However, more information will need to be exposed to support interoperability across a greater diversity of data and tools that are expected to be part of caBIG. We found that data services from Globus are an easy way to implement grid service metadata. The prototype used the native caDSR ISO11179 format as its source. A format such as RDF might also be considered as a possible standard for source metadata in caBIG.
4. **A common query language is important to support federated queries.** The caBIO object model provides numerous objects that model the cancer biology domain. To query the caBIO data grid service, we integrated the object oriented mechanism for issuing caBIO queries into OGSA-DAI using the perform document approach. The caBIO APIs therefore potentially provide the basis for a generalized query syntax across all caBIG data services that are represented through an object model. However, a good deal more analysis is needed before a full-featured common query language can be chosen for caBIG. This will be an essential feature necessary to achieve syntactic interoperability.
5. **Object to data mapping process requires work by data providers**. The caGRID team notes that caBIG data providers must pay particular attention to the problem of mapping data to object models. The caBIO project uses the Apache OJB package for this purpose; another alternative is available from a project called Hibernate. In this prototype, we successfully used the same object model to represent two different database sources operating at physically distinct locations, demonstrating the utility of this object modeling approach in achieving a certain level of interoperability. This is nonetheless a significant additional requirement for sites that are not accustomed to portraying their data in this fashion. Using UML to represent object models allows for implementation in any object-oriented programming language, and thus implementations in languages other than Java are also conceivable. An object-to-data mapping facility will be needed for most implementations in caBIG.

6. **Service description and ontology mapping may benefit from emerging semantic web technologies.** We believe that a rich semantic layer, implemented on top of well-defined object models, will allow users of grid technology to take full advantage of a virtual organization. However, some level of flexibility around the adoption of models may be needed to support the diversity and rapidly changing data and data class relationships supported at different sites. Therefore, requirements for semantic interoperability will need to be considered at multiple levels of granularity: whole models; individual data classes; data class relationships; data elements; data values and value codes. A semantic mapping facility would permit disparate grid-enabled systems to interoperate even if the whole models representing their data are not identical. Structured ontologies represented in semantic web formats such as OWL could provide the anchorage for such a mapping capability.

7 Prospects for informing caBIG

The results of the caGRID prototype project yielded several important lessons that are relevant to the caBIG project.

1. The fundamental goal of the caBIG program is to connect researchers at different geographic locations to semantically interoperable information resources that may be physically located anywhere in the nation's cancer research infrastructure. The grid technology used in this caGRID prototype appears to have the necessary functionality to contribute substantially to the achievement of that goal. We were able to successfully register, advertise, instantiate, and query data services and their corresponding metadata.

2. The OGSA-DAI toolkit provides for an XML-based query interface. This interface should provide for a relatively accessible mechanism for informaticians to interact with the grid. However, the syntactic details within XML still need to be standardized across caBIG in order to provide a truly common query syntax across caBIG grid resources.

3. We were able to extend the framework to support additional types of metadata beyond the core metadata supported by Globus and OGSA-DAI. This was a critical success, since we believed that rich metadata profile would be essential for achieving semantic interoperability in caBIG. The caBIG community will need to identify the complete set of descriptive metadata needed to fully describe and qualify data and analytical grid resources.

4. Patient and other sensitive clinical data may become necessary to transmit across the grid. This will drive the need for proper security procedures and infrastructure. We have identified Globus security framework as a potential place to start when looking for technologies to meet the needs of the cancer community. However, the security features of Globus were not studied in detail and were not implemented in the prototype.

5. We made use of the NCI's Enterprise Vocabulary Resources for looking up synonyms of the names of data classes. However, there are prospects for a much more sophisticated use of semantic web-type data structures to enable semantically rich connectivity and navigability of caBIG resources.

8 Conclusions

We presented a prototype system capable of advertisement, discovery, and query of a five node virtual organization that contains data from disparate sources. These data sources utilized common data elements and APIs, alleviating the need for more elaborate semantic mapping between objects. This multi-site federated data repository allowed researchers to discover clinical trial, genetic, transcript, and agent information around a particular disease or clinical trial phase. We argue that no system is currently capable of providing such information in an automated, standards compliant way. The prototype system uses emerging standards from the OGSA and Globus community to perform advertisement, discovery and query into a virtual organization. Queries such as, “show me all clinical trials in phase I the agents, diseases, and genes that have been targeted in those trials”, can be performed through a simple command line interface or graphical user interface produced during the course of the implementation phase of this project. However, this query cannot be performed using natural language such as described above but, rather, must still be posed in terms of the technologies used to build the virtual organization (VO). We believe that after the grid community agrees and stabilizes the standards around the grid concept and architecture that queries such as the one described above can and will come to fruition.

After a thorough analysis of the technologies and architectures required to implement a grid, the caGRID team found that most large non-commercial projects are using the Globus toolkit from IBM (GT3). The caGRID team chose to implement the prototype system on top of OGSA-DAI as it satisfied most of the requirements discussed in previous sections of this whitepaper. OGSA-DAI (built upon the Globus Toolkit) is geared toward the production of a data centric grid. OGSA-DAI alleviated the need for the caGRID team to produce protocols, interfaces, APIs and query facilities around data centric services.

Unencumbered by protocol and interface issues, the caGRID team sought first to build data centric, OGSA-DAI compliant services using the caCORE framework as the conduit to biological data. This was accomplished through the reverse engineering of OGSA-DAI at places where extension or augmentation were needed. We found that OGSA-DAI did not provide enough metadata around a particular service to make it biologically accessible (discoverable). Therefore, our fist extension involved producing a metadata facility capable of exposing five types of high-level concepts that could be queried during the discovery process. This extension to the framework did not require a large overhaul to the existing code base. Instead, a schema was produced and registered as part of the original registry service in OGSA-DAI. Second, a data factory was produced that served as a conduit between the caBIO data system and OGSA-DAI. The 3.x version of OGSA-DAI system does not provide facilities to write this kind of extension to the system. Therefore, we read some of the source code that was provided as part of the OGSA-DAI framework and produced caBIO data facilities (factories) as part of the OGSA-DAI system. In the future, we would like to see a system capable of registering custom data facilities as a plug-in architecture in OGSA-DAI.

The caGRID team would like to take this project further by allowing users to map data sources other than caBIO into the grid. Therefore, we concluded that this activity would necessitate the need for an object to database mapping facility, possibly as part of the caGRID toolkit. This toolkit has not currently been produced but would be needed in the near future. The caBIO team is documenting its own engineering process and planning to make the necessary combination of tools available as a development kit. caBIO uses the object-to-relational mapping framework provided by the OJB project from the Apache group to bridge its object layer to back-end databases. Another popular package for this purpose is Hibernate.

Once this process has been completed, producing a facility to query the objects produced should be relatively straight forward (as described in previous sections of this whitepaper). However, to truly provide an interoperable “virtual organization” capable of allowing for a fully federated system, additional semantics will be needed to identify data elements and their semantic relationships. It is the opinion of the caGRID team that this requirement can be satisfied using semantic web technologies such as RDF and OWL, and toolkits to manipulate and query these data formats such as Hewlett Packard’s Jena toolkit. The NCI EVS and caDSR registry resources may be able to fulfill some of these requirements with transformation of their content into RDF or OWL as appropriate.

Over the course of the implementation phase of the caGRID project, the team found that the landscape of grid computing changed on a regular basis. These changes included the jargon used, specifications produced, implementations (code base churn), and best practices used. It is this team’s recommendation that users of grid technology try to insulate themselves from API and code base churn as much as they can through careful testing and defensive coding techniques (Design by interface and so forth). This will alleviate tensions between internal requirements and a changing specification.

9 References

1. caGRID project chapter documents

2. caGRID Technology evaluation documents

3. "caCORE: The NCICB Cancer Informatics Infrastructure Backbone", http://ncicb.nci.nih.gov/core

4. "Globus Toolkit 3." http://www.globus.org/ogsa/

5. http://www.ogsadai.org

6. http://www.npaci.edu/DICE/SRB

7. http://www.avaki.com

8. Foster, C. Kesselman, and S. Tuecke, "The Physiology of the Grid" Grid Computing: Making the Global Infrastructure a Reality," F. Berman and A.J.G. Hey, eds., John Wiley & Sons, 2003.

9. The Grid: An Application of the Semantic Web by Carole Goble and David De Roure, October 2002 (preprint)(HTML version). Appeared in Volume 31, Number 4, and December 2002.

10. Covitz PA, Hartel F, Schaefer C, De Coronado S, Fragoso G, Sahni H, Gustafson S, and Buetow KH. caCORE: A common infrastructure for cancer informatics. *Bioinformatics* (2003) 19: 2402-12

11. http://ncicb.nci.nih.gov/NCICB/core/EVS/

12. http://smweb.bcgsc.bc.ca/chinook/

13. http://www.bcgsc.ca/bioinfo/software/discoveryspace/

14. "moby", http://biomoby.org/

15. J. Joseph, C. Fallenstein, "Grid Computing", IBM Press, 2004. p.13

16. I. Foster, C. Kesselman, "The Grid 2", Elsevier, 2004.p.66, 593

17. National Cancer Institute. "caCORE 2.0 Technical Guide", ftp://ftp1.nci.nih.gov/pub/cacore/caCORE2.0_Tech_Guide.pdf

18. Globus Toolkit 3 Core – *A grid service container framework*

19. "Global Grid Forum" 2004 http://www.gridforum.org/

20. OGSA-DAI Product overview p.10,14

21. National Cancer Instiitute. *The NCICB User Applications Manual*.

22. *Globus Toolkit 3 Core – A grid service container framework p.3,6-7*

23. OGSA-DAI Product overview p.10

24. OGSA-DAI-USER-RELASE-NOTES p.4

25. Foster, et al., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of High Performance Computing Applications, vol. 15, 2001.

26. Foster, et al., "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.,"Open Grid Service Infrastructure WG, GGF 2002.

27. "Apache." http://xml.apache.org/

28. N. Hardman, et al., "OGSA-DAI: A look under the hood: Part 2: Activities and results," 2004. http://www-106.ibm.com/developerworks/grid/library/gr-ogsadai2/

29. N. Hardman, et al., "OGSA-DAI: A look under the hood: Part 1: Architecture and database," 2004. http://www-106.ibm.com/developerworks/grid/library/gr-ogsadai2/

30. G Hicken, er al., "How to write an Activity", 2003. http://www.ogsadai.org.uk/docs/R3.1/html/OGSA-DAI-USER-UG-ACTIVITY.htm

31. Krause, T Sugden, A. Borley, "Grid Data Service", 2003, http://www.ogsadai.org.uk/docs/R3.1/html/OGSA-DAI-USER-UG-GDS.htm

32. M. Jackson, M. Antonioletti, A. Krause, "Grid data service factory", 2003, http://www.ogsadai.org.uk/docs/R3.1/html/OGSA-DAI-USER-UG-GDSF.htm

33. M. Jackson, "DAI Service Group Registry", 2003, http://www.ogsadai.org.uk/docs/R3.1/html/OGSA-DAI-USER-UG-DAISGR.htm

34. Anjomshoaa, "OGSA-DAI Graphical demonstrator", 2003, http://www.ogsadai.org.uk/docs/R3.1/html/OGSA-DAI-USER-UG-DEMO.htm

35. M. Antonioletti, A. Anjomshoaa, M. Jackson, N. Chue Hong, "OGSA-DAI Glosary", 2003, http://www.ogsadai.org.uk/docs/R3.1/html/OGSA-DAI-USER-UG-GLOSSARY.htm

36. "DQP" http://www.ogsadai.org.uk/dqp/

37. G. Antoniou, F. Van Harlen, "A semantic web premier", The MIT Press 2004.

38. B. Fenser (E), J. Hedler (E), H. Lieberman (E), W. Wahlster (E) "Spinning the semantic web". 2004.

39. "The NCI center for Bioinformatics", http://ncicb.nci.nih.gov

40. "SRB", http://www.npaci.edu/DICE/SRB/index.html

41. "Polar: Distributed Query Processing on the Grid", http://www.ncl.ac.uk/polarstar/overview.htm

42. "JXTA", www.jxta.org

43. "Avaki", http://www.avaki.com/

44. "ODD-Genes", http://www.epcc.ed.ac.uk/oddgenes/

45. "The data grid project", http://edg-wp2.web.cern.ch/edg-wp2/

46. "e-Science", http://www.escience-grid.org.uk/index.htm

47. "myGrid", http://www.mygrid.org.uk

48. "Birn", http://www.loni.ucla.edu/BIRN/About_BIRN/

49. "International virtual data grid laboratory", http://www.ivdgl.org/

50. S. Malaika, D. Hain, Accessing DB2 Universal Database using the Globus Toolkit and OGSA-DAI, 2003.

51. W. Sanchez, caGRID Project Chapter, 2004.

52. a Foster, C. Kesselman, and S. Tuecke, “The Physiology of the Grid” Grid Computing: Making the

53. Global Infrastructure a Reality," F. Berman and A.J.G. Hey, eds., John Wiley & Sons, 2003.

54. The Grid: An Application of the Semantic Web by Carole Goble and David De Roure, October 2002 (preprint)(HTML version). Appeared in Volume 31, Number 4, and December 2002.

55. "Chinook P2P Bioinformatics", http://smweb.bcgsc.bc.ca/chinook/

56. DiscoverySpace, http://www.bcgsc.ca/bioinfo/software/discoveryspace/

57. "Jena 2 - A Semantic Web Framework", http://www.hpl.hp.com/semweb/jena.htm

58. J. Golbeck, G. Fragoso, F. Hartel2, J. Hendler, J. Oberthaler, B. Parsia, The National Cancer Institute's Thesaurus and Ontology, 2003, http://www.mindswap.org/papers/WebSemantics-NCI.pdf

Appendix A – Supplementary Requirements

Requirement	Globus / OGSA-DAI / DQP	SRB	Web Services	Jena
1. Functionality				
Discovery: Ability to discover local and/or remote data source available.	Yes, using information services. OGSA-DAI & DQP provide directory services as well.	Yes. All registered and accessible data sources are listed in the MCAT server.	Yes. Through Universal Description, Discovery and Integration (UDDI)	NA
Data sharing: Ability to share data beyond the company firewall. The system shall advertise data sources.	Yes, most implementation is SOAP on HTTP.	Yes. Multiple clients available using various protocols.	Yes. See Globus.	NA
Virtual organizations: Ability to create virtual organizations.	Yes, aggregating and indexing services.	MCAT server can support one or more domains.	Yes. With custom implementation.	NA
Life cycle: The system must support service life cycle. For example, advertise data source for one week.	Yes, services can be created with lifetime.	No. Service type implementation is not supported.	Yes. Built upon Servlet technology.	NA
State management: The system shall provide service state at any given time.	Yes, using default services or implementing additional data services.	Limited feedback through command line tools.	Yes. Built upon Servlet technology.	NA
Creation and destruction – factory: The system shall have the ability to create and destroy service at any time.	Yes, using default services and port types.	No. Not a service type implementation.	Yes	NA
Event notification: The system shall log/notify events.	Yes, core and information service modules.	Yes. Limited logging is enabled.	Yes	Yes
Add/Delete Researchers: The system shall add/delete researchers for one or many virtual organizations.	Yes	Yes	No. This concept does not exist.	NA
Information services: Information about the resources available on the system (Grid) should be accessible through information services. This information should be automatically maintained and up to date.	Yes, with data services and information services.	Yes, information is stored in the MCAT server.	Yes, through UDDI.	NA
Resource Brokering: Users should submit their requests to a resource broker specifying their high level requirements. The Resource Broker should be able to find and allocate suitable resources by querying information services.	Yes, it could be configured in different ways. Aggregating and indexing services.	Batch jobs for Data ingestion through SRB.	Yes, with custom implementation.	NA
Uniform access to resources: All the resources of the same kind (computing elements, storage elements, etc.) should be accessed in a uniform way, no matter which technologies or standards they are based on. This should be done through software modules installed on each single system that hide heterogeneity and provide uniform interfaces (for example, APIs).	Yes, strong in this framework.	Yes. Actual data source is hidden through one of the many client interfaces.	NA	NA
Data Access: Users should be able to access distributed data in a uniform fashion.	Yes	Yes. Data access layer is abstracted.	No	NA
Data Replication: The system (grids) should allow automatic file replica creation in order to move data closer to the user or to the	Not with details services, however it can be implemented. Not	Yes. One or more SRB servers can remain in sync. But data replication	No	NA

Requirement	Globus / OGSA-DAI / DQP	SRB	Web Services	Jena
computing facilities that will process them.	included in the evaluation.	in one server is not yet supported. Will have to rely on database level replication.		
The system should support that the user nodes are not available all the time. The system should be aware and manage service status accordingly.	Yes.	Yes. A user node can be synched with the SRB server when it connects.	Yes, with UDDI.	NA
2. Security				
	Globus comes with a security component; however it was not included in the evaluation.			
Multiple infrastructures	Not tested	Not Tested	Not tested	NA
Application and network firewalls	Not tested	TCP/IP	Yes	Yes
Distributed trust mechanism	Not tested	Yes, federated MCAT servers.	Not tested	NA
Reputation	Not tested	Not Tested	Not tested	NA
Sand-boxing	Not tested	Not Tested	Not tested	NA
Anonymity	Yes	Guest logins are possible.	Not tested	NA
Provide robust, behind-the-scenes services such as authorization, access control, and delegation.	Not tested	GSI or other certification scheme can be implemented.	Yes	NA
Provide security mechanisms that enable system administrators to enforce access rules for all the resources made available on the system.	Not tested	Yes	Yes	NA
Use of X.509 certificates and proxy delegation allows systems to verify grid users' identity without exposing their credentials on the Internet.	Not tested	Yes	Yes	NA
Support authentication.	Not tested	Yes	Yes	NA
Support authorization.	Not tested	Yes	Yes	NA
Support both secure and insecure virtual organizations.	Not tested	Yes. Guest proxy based access.	Not tested	NA
Use of encryption preserves confidentiality.	Not tested	No	Yes	NA
3. Scalability				
	Grid infrastructures can grow as the requirements grow. Need additional stress test.	Multiple servers/data sources can be seen as one logical resource.		
4. Resource management				
CPU scavenging	Not tested	Not Tested	Not tested	NA
Load balancing	Not tested	Multi threaded data exchange	Not tested	NA
5. System properties				
Fault tolerance	Not tested	No	Not tested	NA
Self-healing	Not tested	No. It relies on database durability to deal with crashes and runtime errors.	Not tested	NA

Requirement	Globus / OGSA-DAI / DQP	SRB	Web Services	Jena
6. Group support				
Virtual organization concept	Yes	User groups are divided into domains and zones.	No	NA
7. Monitoring				
Job scheduling - Jobs submitted by the users should be effectively scheduled.	Not tested	*Not tested but possible*	*Not tested*	*NA*
8. Connectivity				
Traversing firewalls	Yes, mostly HTTP	Through web client	Yes	NA
Dynamic IP addresses and Network Address Translations (NATs)	NA	NA	Yes	NA
Network characteristics are different	NA	NA	NA	NA
Temporal disconnects	Yes for clients. Need additional testing for servers.	NA	Yes	NA
Ad-hoc networks (APPAGG-RG)	NA	NA	NA	NA
9. Decentralization				
Fault tolerance might be related and self-healing are related (auto reconfiguration).	Not tested	No	Yes	NA
10. Even servers crash				
Decentralized infrastructure service	Yes	Different configurations possible with the federated MCAT server.	Yes	NA
Dynamic entry/exit	__		NA	NA
11. Quality of Services				
Data quality and corruption	Yes	Using Checksum	Yes	Yes
12. Industry / NCI standards:				
Must be P2P compliant	NA	Works through a central MCAT server.	Yes	Yes
Must be grid computing compliant	Yes	Yes, GSI compliant.	No	No
Leverage existing solutions where appropriate	Legacy application integration with minor changes.	Legacy data sources can be used with existing or custom drivers.	Yes	Yes
Standard interface protocols	Yes, industry and NCI, NCICB and caBIG standards.	Yes, TCP, HTTP and NetBIOS.	Yes	NA
Use HTTP as a transport protocol	Yes	Only through the web client.	Yes	Yes
Have easily-understandable programming model	Yes, learning curve middle to high.	Java API	Yes	Yes
Perform parallel programming	Yes, but not tested.	Optimized algorithms for data access	Yes	NA
The system is JMS compliant.	Yes, it is part of the implementation.	No	Yes (see Apache Axis implementation)	NA
Web services	Yes, all services are web services.	No	Yes	Yes
Open source	Yes, Globus Alliance, eScience, my GRID, Axis, Tomcat, Java and C++.	Licensed	Yes	Yes

Requirement	Globus / OGSA-DAI / DQP	SRB	Web Services	Jena
J2EE Complaint	Yes	C source with Python bindings and Java API	Yes	NA
Java source code	Yes	C source	Yes	Yes
Local Data source is JDBC compliant	Yes	OCI	Yes	Yes
13. Semantic				
Well defined API	Yes	Yes, Limited API.	Yes	Yes
Ability to use relational backend	Yes	Yes	Yes	Yes
Ability to read/write/modify OWL/RDF/DAML-OIL ontology language	No	NA	No	Yes
Ability to read/write/modify RDFS/XML representation of semantic objects	No	NA	No	Yes
Ability to define ontologies on the fly	No	NA	No	Yes
Ability to map ontologies to one another	No	NA	No	Yes
Reification ready	No	NA	No	Yes
Inference engine capability	Yes	NA	No	Yes
RDQL aware/ready/plug-in	No	NA	No	Yes
Ability to find intersection/union of two related ontologies	No	NA	No	Yes

Appendix B – caGRID Project Use Cases

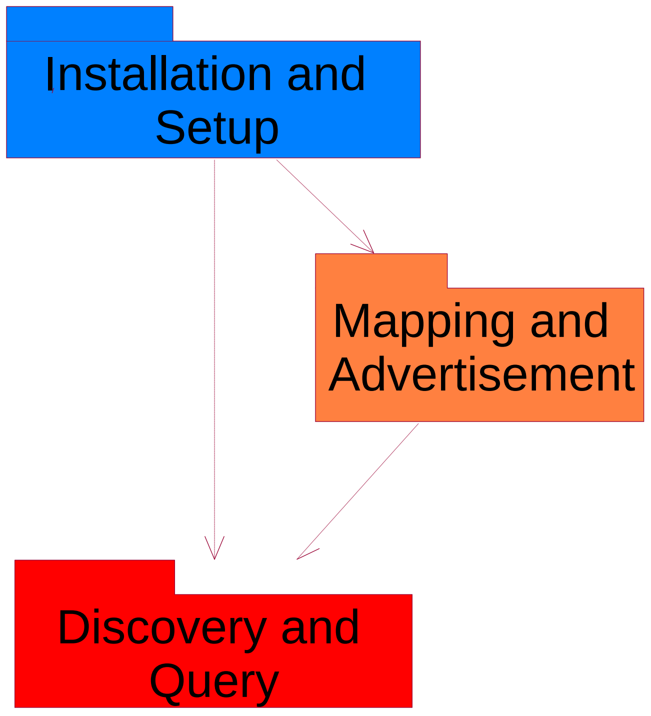


Figure B-1: Use Case Overview

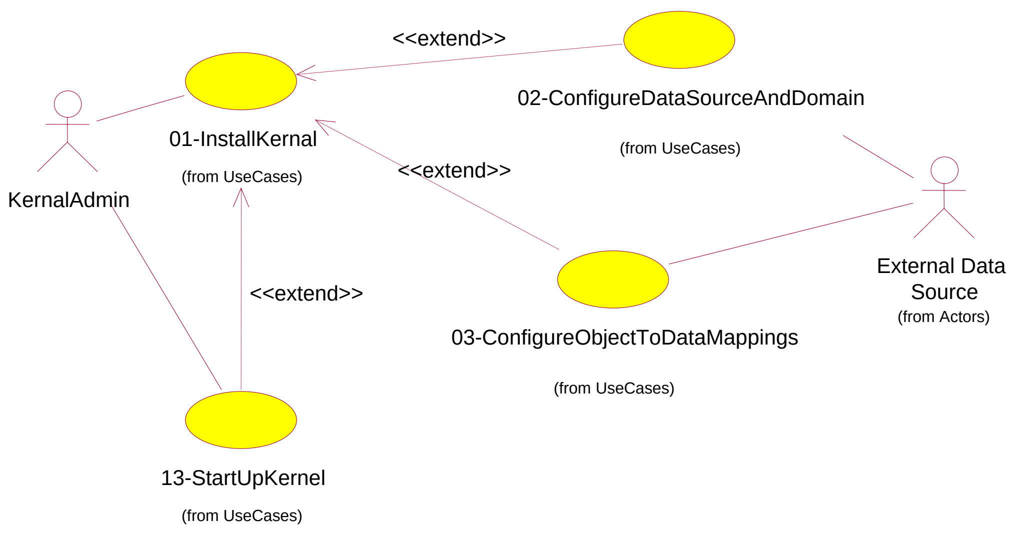


Figure B-2: Install and Setup Use Case

Figure B-2 includes the following use cases:

2.1 Install caGRID
2.2 Configure data source and domain APIs
2.3 Configure object to data mappings
2.4 Startup caGRID
2.5 Setup privileges
2.6 Shutdown caGRID
2.7 Login admin tool

2.1 Install caGRID

Description:

This use case describes caGRID installation on the user computer. The user has different options to install caGRID. The simplest option is to download and install the software without starting up caGRID.

Actors: caGRID administrator.

Extended use case: Startup caGRID, configure data source and domain, configure object to data mapping.

Pre-Conditions:

- Computer with Internet access
- Local cache empty

Basic Course:

1. The use case begins when the actor selects to install caGRID software from http://cabio.nci.nih.gov web site.
2. The system shows a form explaining the download and installation process. To download caGRID, the system provides two options. The first one is to download only caGRID and the second option is to download the caGRID and the Java software.
3. The actor selects to download only caGRID.
4. The system asks for the location to download the jar file.
5. The actor writes down the path.
6. The system downloads the installation packages in the path specified.
7. The actor opens the installation package.
8. The system verifies the minimum software requirements.
9. The system asks for the member name within the virtual organization.
10. The actor enters the name.
11. The system asks if it uses a proxy server.
12. The actor enters "No" for proxy server.
13. The system asks the actor to enter user name, password and password verification.
14. The user enters the user name and password, and confirms the installation.
15. The system asks if the actor wants to startup the caGRID service at the end of installation process.
16. The actor answers "Yes".
17. The system asks if the actor wants to configure data source and object to data mapping.

18. The user answers "Yes" to both questions and the use case ends.

Alternative flows:

Not all required software installed:

a. On step 8, the system cannot install the software because all required software is not properly installed and configured. The system displays a message indicating the issue.
b. The user confirms the message and the use case ends.

Cannot startup the caGRID:

a. On step 16, the system cannot startup caGRID. The system displays a message indicating the issue.
b. The user confirms the message and the use case ends.

Firewall configuration:

a. On step 12, the actor enters yes for proxy server.
b. The system asks for proxy address and port number.
c. The actor enters the proxy address and port number.
d. The system asks for the address of the bridge between the internal peers and the peers on the Internet.
e. The user enters the bridge address and the port number.
f. The use case continues on step 13.

Java and other software installation:

a. On step 3, the actor downloads the required software such as Java along with the caGRID.
b. The system shows the list of the software to install along with the link for detailed installation instructions.
c. The user installs and verifies the new software installation.
d. The use case continues on step 4.

Post Conditions:

1. If there were no errors in the process, caGRID was installed properly.

2.2. Configure Data Source And Domain APIs

Description:

This Use Case describes how to configure caGRID (either locally or at the site level). This use case includes configuring new data sources and new domains to a local caGRID.

Extension point: This use case extends from the "Install caGRID" use case step 18 from the basic course of events.

Assumptions:

- System marks the first data source created as default.

Actors: caGRID administrators, Scientist

Pre-Conditions:

- Available Domain APIs (such as caBIO) configuration gets created (from caGRID Installation Use Case)

Basic Course:

This use case begins when the caGRID administrator or scientist wants to configure a data source or domain. The system displays a screen, which has the functionality available to that user. In this section the user can:

a) Configure a data source.

b) Configure a Domain.

Sub flows:

Configure data source:

a. Actor configures ODBC Data Sources (for example, through control panel) as required. Actor provides detailed parameters like driver, name and description.

b. Actor configures JDBC Data Source by providing JDBC URL, JDBC Driver Class Name, data base Login, and Drivers jar file location.

c. Actor will be able to specify default JDBC Data Source.

d. The system persists the Data Source configurations and the use case ends.

Configure domain:

1. System displays list of already pre-configured Domain APIs (from *caGRID Installation* Use Case). The system displays a form to create the new domain.
2. Actor provides the name and location of the source (probably a jar file).
3. System displays the services.
4. The actor selects one of the services.
5. The system displays a property page of the services.
6. The actor updates the service properties and submits the changes.
7. The system persists the changes and the use case ends.

Alternative flows:

Cannot persist domain or data source:

a. On step 4 (from configure data source section) or on step 7 (from configure domain section), the system cannot persist the new domain. The system displays a message indicating the issue.
b. The user confirms the message and the use case ends.

Post Conditions:

1. If there were no errors in the process, a new data source or domain was created.

2.3 Configure Object To Data Mappings

Description:

This use case describes how to map an Object model such as caBIO with a local Data Source at the metadata level. The mapping process is between one object-attribute from the object model with one field-table from the data source. After the actor maps the data, the system persists these mappings. caGRID uses these mappings during local data retrieval. Object to data mapping can be launched from the installation process or from the caGRID admin tool.

Extended use case: Setup EVS mapping, Setup manual NCICB mapping, advertise data source.

Assumptions:

- One object-attribute to one table-column is only supported.

Actors: Administrator, Scientist

Pre-Conditions:

- At least one data source is configured in caGRID.
- The caGRID user has privileges to data source and domain.
- Actor has launched the Admin tool.

Basic Course:

1. This use case begins when the user selects the “Configure object to data mapping” option.
2. System displays available Domain APIs.
3. Actor selects an API such as caBIO.
4. System displays available Data Sources from the domain selected on previous step.
5. Actor selects a Data source.
6. System displays all available tables for data mapping.
7. For each table:
   7.1 Actor selects a table from the data source and a field from the table.
   7.2 Actor selects an object (such as Gene) from the Domain and an attribute from the object.
   7.3 Actor maps the selected Object-attribute with a Table-field.
   7.4 Actor selects an Object from the domain (such as Gene) and an attribute for mapping.
   7.5 Actor selects option to persist the mapping and assign a name and description to the mapping.

7.6 System persists Object-To-Data Mappings locally.

7.7 Actor optionally advertises the new service.

7.8 End for.

8. The use case ends.

Alternative flows:

No available Domain APIs:

a. On step 2, the system does not find Domain APIs. The system displays a message that there are no available Domains.
b. The user confirms the message and the use case ends.

No available data sources:

a. On step 4, the system does not find Data sources. The system displays a message that there are no available data sources.
b. The user confirms the message and the use case ends.

No privileges for database object:

a. On step 6, when the user attempts to look for the tables, the database rejects access and displays an access violation message.
b. The user confirms the message and the use case ends.

Cannot persist the mapping:

a. On step 7.6, the system cannot persist the new mapping. The system displays a message that the mapping could not be persisted
b. The user confirms the message and the use case ends.

Post Conditions:

1. If the process was successful, at least one Object-To-Data Mapping is persisted locally.

2.4 Startup caGRID

Description:

This use case describes caGRID startup process triggered either by the operating system or by the caGRID user. During the caGRID installation, the user can select to startup the service automatically or manually. When automatic is selected, caGRID is launched when the system boots up. When manual is selected, the caGRID is launched by the user.

Extension point: This use case in extended from the "Install caGRID" use case at step 16.

Actors: caGRID administrator, scientist, host system

Pre-Conditions:

- caGRID installed
- User with startup privileges

Basic Course:

1. The use case begins when the actor selects caGRID startup, or when the system boots up. The user can startup caGRID as part of the caGRID installation or any time after. The system starts up caGRID when the caGRID properties have automatic startup activated.
2. The system opens the properties file, verifies the user/system credential and starts caGRID accordingly. The system identifies that a firewall is not required.
3. The system launches the discovery process. Include discovery use case and the use case ends.

Alternative flows:

Not all required software installed:

a. On step 2, the system identifies that a firewall is required, the system references the bridge and firewall address for future use in the discovery process and the use case continues on step 3.

Cannot startup the caGRID:

a. On step 2, the system cannot startup caGRID. The system displays a message indicating the issue.
b. The user confirms the message and the use case ends.

Post Conditions:

- caGRID initiated.

2.5 Setup Access Privileges

Description:

This use case describes the configuration and functionality of the Security component involved in caGRID. The security deals with who can access what data.

Assumptions:

1. By default, the local data will not be shared with other caGRID systems.
2. Data sharing will be limited to entire site level only. The entire site data (for which Object-To-Data mapping has been done) will be shared.
3. Data will be shared with only specified Data Source Indicators (DSIs).

Actors: caGRID administrator

Pre-Conditions:

- Advertisements for Domain Objects were already published

Course of Action:

1. Actor selects to configure access privileges.
2. System, using Discovery Service API, displays all DSIs available.
3. Actor specifies what DSIs can see the local data.
4. System updates local Advertisements for Domain Objects with the list of DSIs that can view the local data.

Post Conditions:

- Advertisement Service API is available

Issues:

- None

2.6 Shutdown caGRID

Description:

This use case describes the caGRID shutdown process triggered either by the operating system or by the caGRID user.

Actors: caGRID administrator, scientist, host system

Pre-Conditions:

- caGRID running
- User with shutdown privileges

Basic Course:

1. The use case begins when the actor selects caGRID shutdown, or when the system shuts down.
2. The system persist the cached data.
3. The system closes the caGRID and the use case ends.

Alternative flows:

Cannot persist data:

a. On step 2, the system cannot persist cached data. The system displays a message indicating the issue.
b. The user confirms the message and the use case ends.

Post Conditions:

- caGRID not available

2.7 Login admin tool

Description:

This use case describes the caGRID login admin tool GUI process.

Actors: caGRID administrator, scientist

Pre-Conditions: NA

Basic Course:

1. The use case begins when the actor selects caGRID GUI tool.
2. The system asks for the username and password.
3. The actor enters user name and password.
4. System validates actor credential, starts the admin tool, and the use case ends.

Alternative flows:

User name / password not valid:

a. On step 4, the system cannot recognize the user. The system displays a message indicating the issue.
b. The user confirms the message and the use case ends.

Not enough privileges:

a. On step 4, the system cannot open the GUI tool because the user does not have privileges to run the tool. The system displays a message indicating the issue.
b. The user confirms the message and the use case ends.

Post Conditions:

- caGRID not available

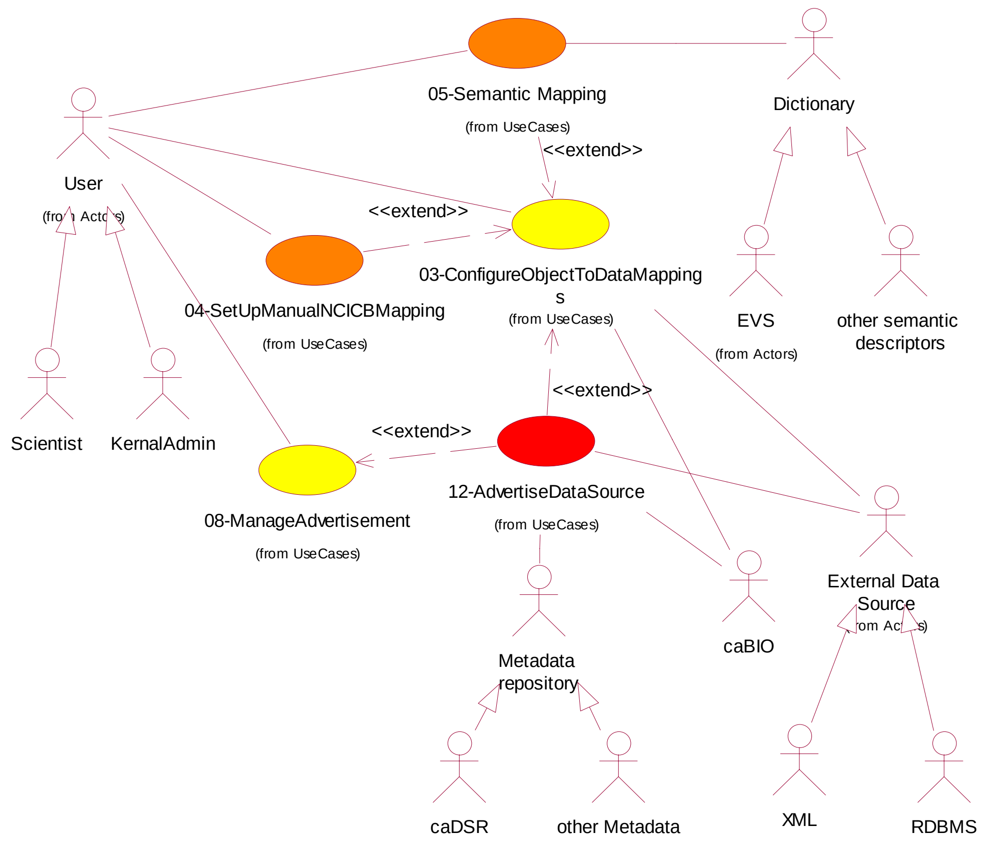


Figure B-3: Management and Advertisement Use Case

Figure B-3 includes the following use cases:

3.1 Setup manual NCICB mapping

3.2 Setup EVS mapping

3.3 Advertise services

3.4 Advertise data source

3.1 Setup manual NCICB Mapping

Description:

This use case describes how an actor performs manual mapping of local data with NCICB data (at the data level). The system persists data these mappings. caGRID uses these mappings to submit the query to other DSIs.

Extension point: This use case is extended from the "Configure object to data mapping" use case at step 7.6.

Assumptions: None

Extended use case:

Configure object to data mapping.

Actors: caGRID administrators, Scientist

Pre-Conditions:

- Actor is already logged into caGRID administration tool.

Basic Course:

1. The use case begins when caGRID administrator or scientist selects Setup manual NCICB mapping from the caGRID admin tool. The use case can also begin when the actor is creating a new object to data mapping, and as soon as the mapping is ready, the actor optionally can set up the NCICB mapping.
2. System displays available Data Sources for which Object-To-Data Mappings have been configured.
3. Actor selects a Data Source for manual mapping.
4. System displays list of all table-columns in the selected Data Source for which there exists entries in the locally persisted Object-To-Data mappings.
5. Actor selects columns in a table.
6. System consults locally persisted Object-to-Data mappings and retrieves values from selected local data source and NCICB data source for the above selected table-columns.
7. System displays local data that is not matched to NCICB data.
8. Actor maps data values between the local and NCICB data sources.
9. System persists manual Data Mappings.

Alternative flows:

No available Data source:

1. On step 2, the system does not find data sources locally. The system displays a message that there is no data source.
2. The user confirms the message and the use case ends.

No table available:

a. On step 4, the system cannot display tables. The system displays a message that there are no tables available.

b. The user confirms the message and the use case ends.

Cannot persist data:

a. On step 9, the system cannot persist data. The system shows a message indicating the problem.

b. The user confirms the message and the use case ends.

Post Conditions:

1. Manual Mapping of Local Data to NCICB Data is persisted

3.2 Setup EVS mapping

Description:

This use case describes how an actor performs Symantec mapping of unmatched local data with Enterprise Vocabulary Service (EVS) data. The system persists these data mappings. caGRID uses these mappings to submit the query to other DSIs.

Extension point: This use case in extended from the "Configure object to data mapping" use case at step 7.6.

Assumptions: None

Extended use case:

Configure object to data mapping.

Actors: **caGRID** Administrator, scientist

Pre-Conditions:

- Actor is already logged in into caGRID administration tool.

Basic Course:

1. The use case begins when caGRID administrator or scientist selects Setup EVS mapping from the caGRID admin tool. The use case can also begins when the actor is in the object to data mapping process, and as soon as the mapping is ready, the actor optionally can set up the EVS mapping.
2. System displays available Data Sources for which Object-To-Data Mappings have been configured.
3. Actor selects a Data Source for Symantec Mapping.
4. System displays list of all table-columns in the selected Data Source where there exists entries in the locally persisted Object-To-Data mappings.
5. Actor selects columns in a table.
6. System consults locally persisted Object-to-Data mappings and retrieves values from selected local data source and NCICB data source for the above selected table-columns.
7. System displays local data that is not matched to NCICB data and the corresponding data from EVS (that matched NCICB data) using EVS APIs.
8. Actor maps data values between the local data and EVS data.
9. System persists Symantec Data Mappings and the use case ends.

Alternative flows:

No available Data source:

1. On step 2, the system does not find data sources locally. The system displays a message that there is no data source.
2. The user confirms the message and the use case ends.

No table available:

a. On step 4, the system cannot display tables. The system displays a message that there are no tables available.

b. The user confirms the message and the use case ends.

Cannot persist data:

a. On step 9, system cannot persist data. The system shows a message indicating it.

b. The user confirms the message and the use case ends.

Post Conditions:

1. Semantic Mapping of Local Data to EVS Data is persisted

3.3 Advertise services

Description:

This use case describes how caGRID at a given site publishes data services. That is, what domain objects is it going to serve and based on what criteria.

Extended use case: Advertise data source

Assumptions:

- caGRID displays the local data sources that the users have privileges to advertise.

Actors: caGRID Administrator, Scientist

Prerequisite Use Cases: NA

Pre-Conditions:

- At least one data source is configured in the caGRID.
- caGRID user has privileges to data source and domain.
- Actor has launched the Admin tool.

Basic Course:

1. This use case begins when the caGRID administrator or scientist wants to advertise a new service, update the properties of a service or delete a service. The system displays a screen, which has the functionality available to that user. In this section the user can:
   a. Advertise a new service.
   b. Modify the properties of a service already advertise.
   c. Delete/Disable a service from caGRID administrator or scientist.

Sub flows:

Advertise data source:

a. System consults locally persisted Object-To-Data Mappings and displays a list of available Domain Objects and mapped attributes.
b. For each data source, the actor optionally selects advertise the data source.
c. The use case ends.

Modify the properties of a service already advertised:

a. System consults for the services already advertised.
b. System displays the services.
c. The actor selects one of the services.
d. The system displays a property page of the services.
e. The actor updates the service properties and submits the changes.
f. The system persists the changes and the use case ends.

Delete/disable a service:

a. System consults for the services already advertised.
b. System displays the services.
c. The actor selects one of the services.
d. The system displays a message asking for confirmation.
e. The actor confirms the action.
f. The system deletes/disables the service and the use case ends.

Alternative flows:

No available Data source:

a. On step 1 from advertise source, the system does not find data sources locally. The system displays a message that there is no data source.
b. The user confirms the message and the use case ends.

No service advertised available locally:

a. On step 1 from Modify and delete a service, the system does not find a service already advertised. The system displays a message that there are no services advertised.
b. The user confirms the message and the use case ends.

Post Conditions:

1. When advertise data source finished successfully, at least one new service is advertised.
2. When modify data source finished successfully, the services selected has persisted the new properties.
3. When disable/delete data source finished successfully, the service selected is no longer available.

3.4 Advertise Data Source

Description:

This use case describes how to advertise one data source. The actor selects the groups (virtual organization) that will use the service and define the life cycle of the service.

Extension point: This use case in extended from "Advertise services" use case step 2 from the advertise data source course of events, and from step 7.7 from the basic course of actions

Assumptions: NA

Pre-Conditions: NA

Course of Action:

1. The use case begins when the actor has selected/created a data set and selects the option advertises.
2. The system displays the list of groups (virtual organizations) available.
3. Actor selects the groups that will see the data source.
4. Actor specifies the life cycle of the service advertised.
5. System advertises the data sources to the virtual organizations selected and the use case ends.

Alternative flows:

No virtual organizations created:

a. On step 2, the system does not find virtual organization. The system displays a message that there is no virtual organization.
b. The user confirms the message and the use case ends.

Advertisement failure:

a. On step 5, the system cannot advertise the services. The system displays a message that an error occurs when trying to advertise data sources.
b. The user confirms the message and the use case ends.

Post Conditions:

1. At least one data source advertises when the course of action finished successfully.

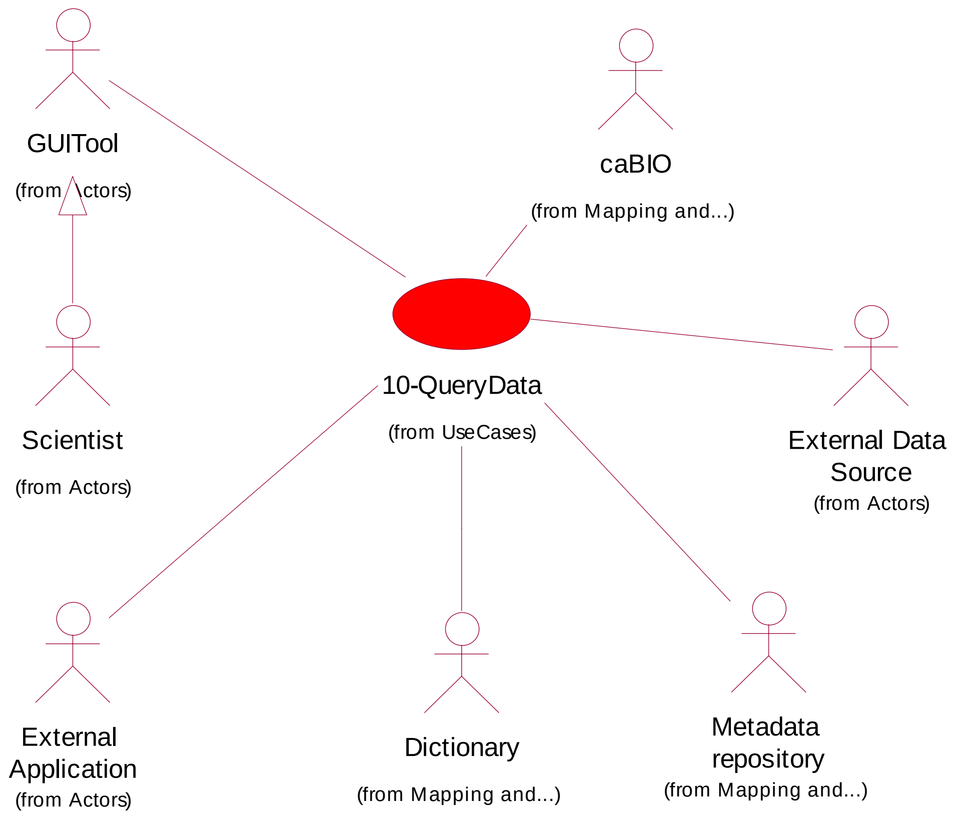


Figure B-4: Discovery and Query Use Case

Figure B-4 includes the following use cases:

4.1 Discovery Service

4.2 Query Data

4.1 Discovery Service

Description:

This use case describes Discovery Service API. Using this API, the client will be able to discover all DSIs and the metadata that they support. Discovery is the process of one peer searching for another peer, in the same peer group, that contains the desired content.

Actors: caGRID, caGRID Client, Local Developer

Pre-requisite Use Cases:

- Advertise Services in metadata Format

Pre-Conditions:

- Advertisements for Domain Objects were already published

Course of Action:

1. Actor makes a Discovery Service API call (supplying a query as input) and retrieves all the DSIs that can serve the query.
2. System queries all caGRID systems and returns DSI list along with metadata.

Post Conditions:

1. Discovery Service API available

Issues:

1. Does Security need to be incorporated

4.2 Query data

Description:

This use case allows the user to retrieve data from multiple Data Source Indicators (DSIs). A Data Source Indicator refers to the data source at a site. These DSIs will be made available through Advertisements/Services to other caBIO servers. The actor has the option to query the system in different ways as follows: Request data sources (local and remote), request object to data mapping, request attributes or request data.

The actor can instantiate the object only once to get the information desired, or all to get the data results beginning with the data sources available.

Actors: GUI Client, Scientist

Basic Course:

1. This use case begins when the actor instantiates the query object. The actor optionally uses the request to get different kind of data. The actor may instantiate the object with one of the following options:
    a. Request data sources (local and remote)
    b. Request object to data mapping
    c. Request attributes
    d. Request data

Sub flows:

Request data sources (local and remote):

a. The actor requests the data source available. The request includes the user identification that includes username, password and VO.
b. The system returns the list of local and remote data sources available. The list includes data source name and data source owner.

Request object to data mapping:

a. The actor requests the object to data mapping for a particular data source. The request signature includes username, password, VO, and data source.
b. The system returns the list of data mappings for the given data source.

Request attributes:

a. The actor requests attributes available for one specific mapping. The signature of the request is the username, password, VO, data source, and object to data mapping.
b. The system validates the request.
c. The system performs the request and returns the attribute list.

Request data:

a. The actor performs a query with the following parameters: username, password, VO, data source, object to data mapping, attribute and filter value. The filter could be more than one option using and/or conditions.
b. The actor submits the query.
c. The system validates the request.
d. The system identifies the location of the query and sends the query to the database.
e. The system receives the results from the database.
f. The system returns the query results.

Alternative flows:

NCI Data mapping:

a. On step 4 from request data sub-flow, the system identifies that there is at least one NCICB data mapping.
b. The system reads the mapping.
c. The system returns the results.

EVS Data mapping:

a. On step 4 from request data sub-flow, the system identifies that there is at least one EVS data mapping.
b. The system sends the request to EVS API indicating username, password, VO, data source and object to data mapping.
c. EVS sends back the mapping results.
d. The system returns the mapping results.

No available Data source:

a. On step 1 form advertise source the system doest not find data sources locally. The system displays a message that there is a not data source.
b. The user confirms the message and the use case ends.

No service advertised available locally:

a. On step 1 from Modify and delete a service, the system does not find a service already advertised. The system displays a message that there are no services advertised.

Post Conditions: NA

Appendix C – Prototype Demo Screen Shots

Below is a graphical view of the prototype test environment we are hosting in conjunction with NCICB and Boston locations. The major difference in these two data centers is the Clinical Trial Data. Panther Informatics, Boston is hosting SPORES Clinical Trial Data. Some of the test scenarios capture this difference effectively.

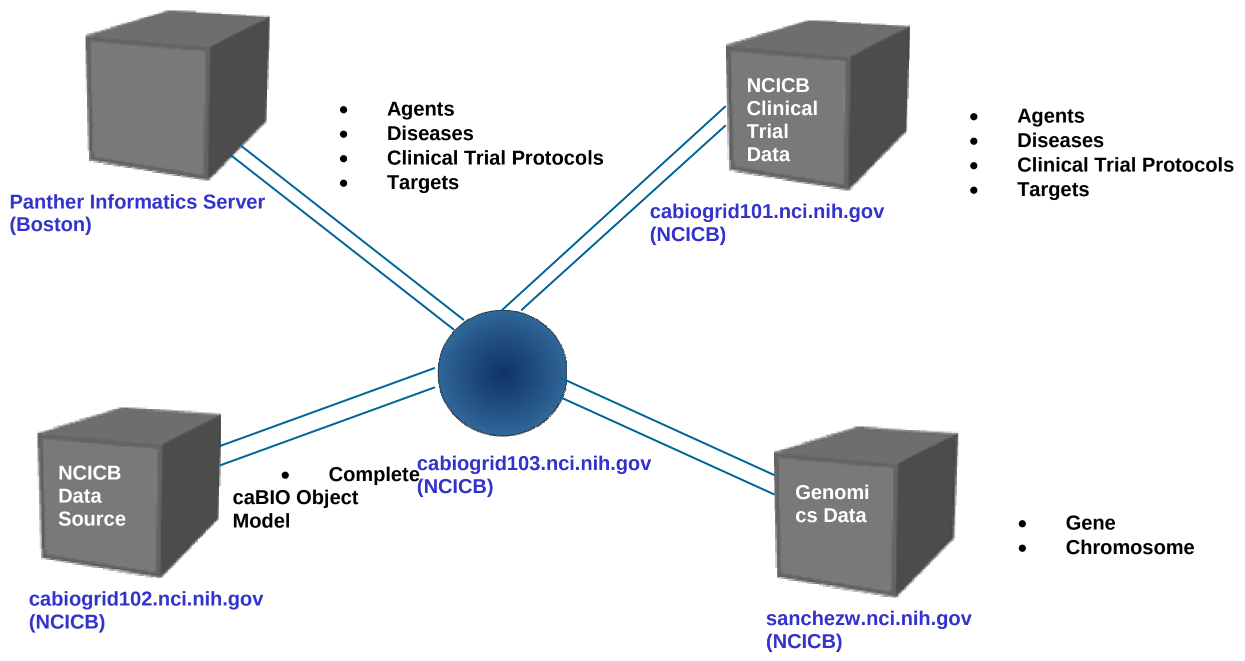
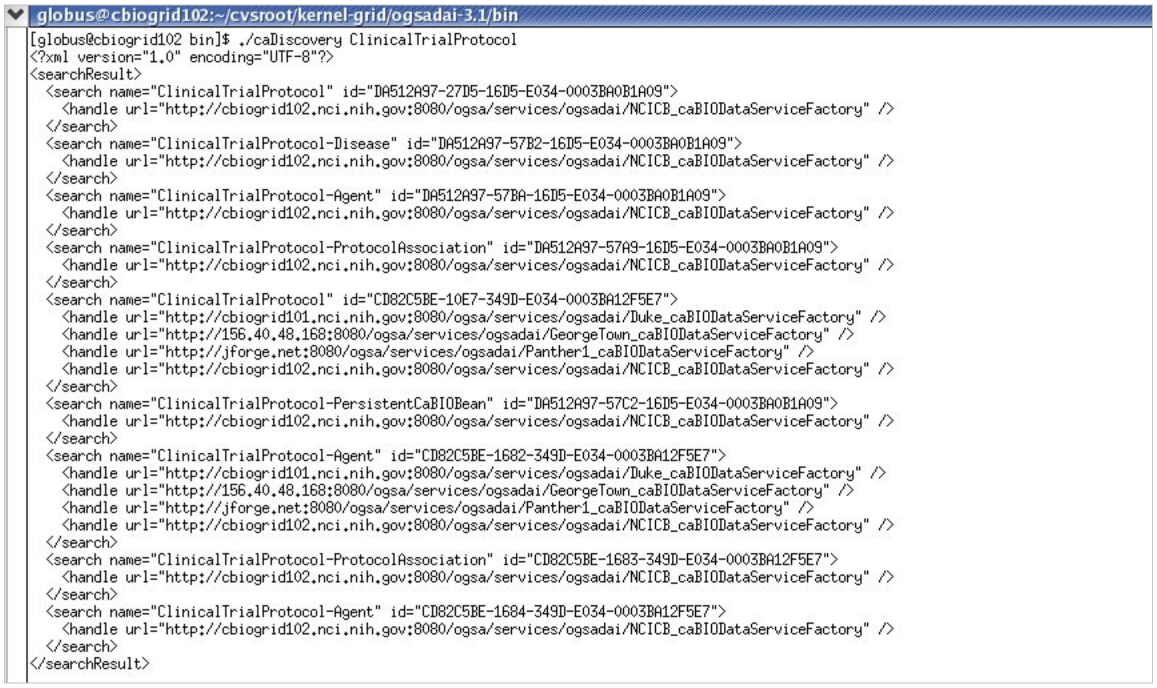

Panther Informatics Server (Boston)
• Agents
• Diseases
• Clinical Trial Protocols
• Targets
NCICB Clinical Trial Data
cabiogrid101.nci.nih.gov (NCICB)
• Agents
• Diseases
• Clinical Trial Protocols
• Targets
NCICB Data Source
cabiogrid102.nci.nih.gov (NCICB)
• Complete caBIO Object Model
cabiogrid103.nci.nih.gov (NCICB)
Genomics Data
sanchezw.nci.nih.gov (NCICB)
• Gene
• Chromosome

Figure C-1: caGRID Prototype Demo Setup

Some of the sample scenarios that can be executed with the caGRID prototype are:

- Search and retrieve Clinical Trial Protocols for Neoplasia
- Search and retrieve Agents (drugs) associated with these trials
- Search and retrieve Diseases that these Agents are associated

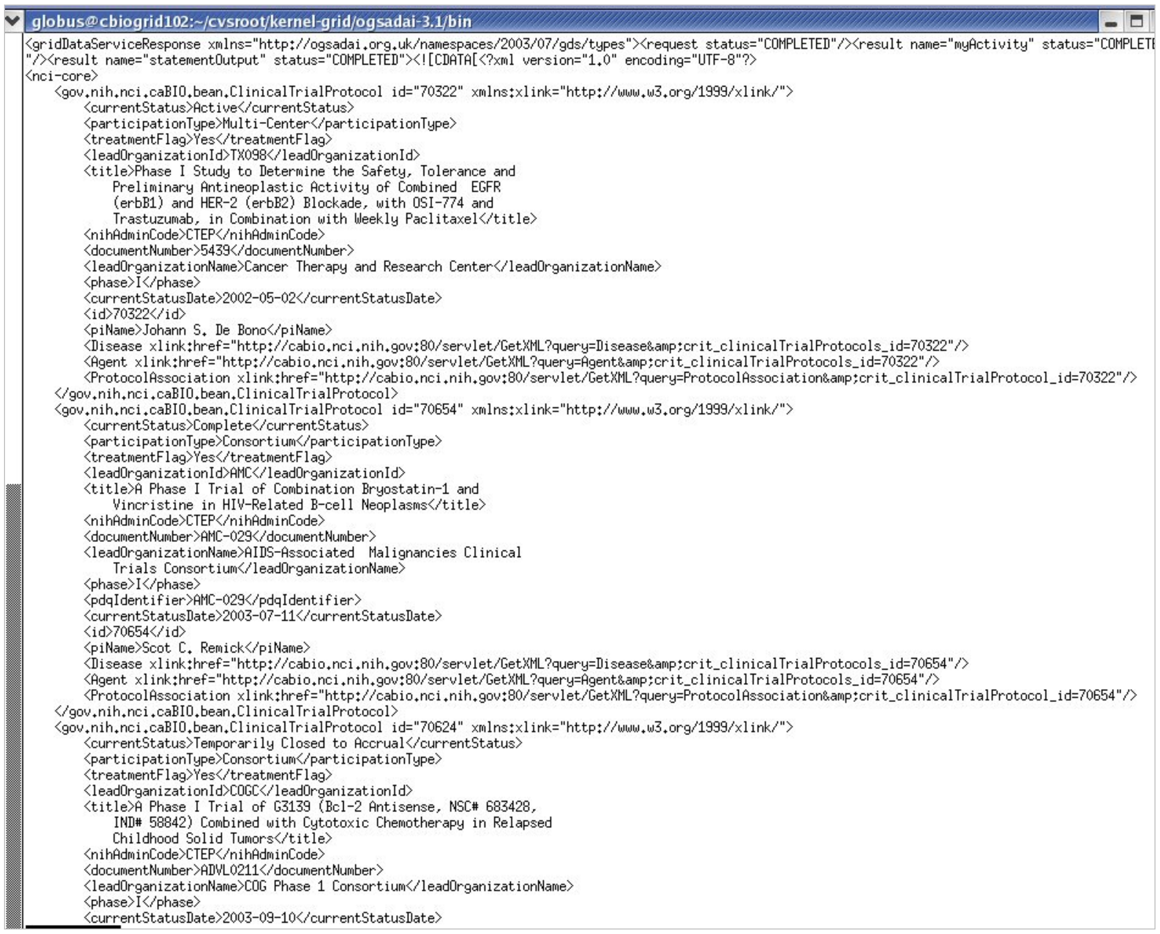
Following are screenshots from a recent demo of the caGRID prototype. There are three client applications that are a combination of Globus Toolkit, OGSA-DAI and in-house development efforts. Several efforts are underway here at NCICB to develop GUI and command line clients to search (Discover) and query data services on the grid.



globus@cbiogrid102:~/cvsroot/kernel-grid/ogsadai-3.1/bin

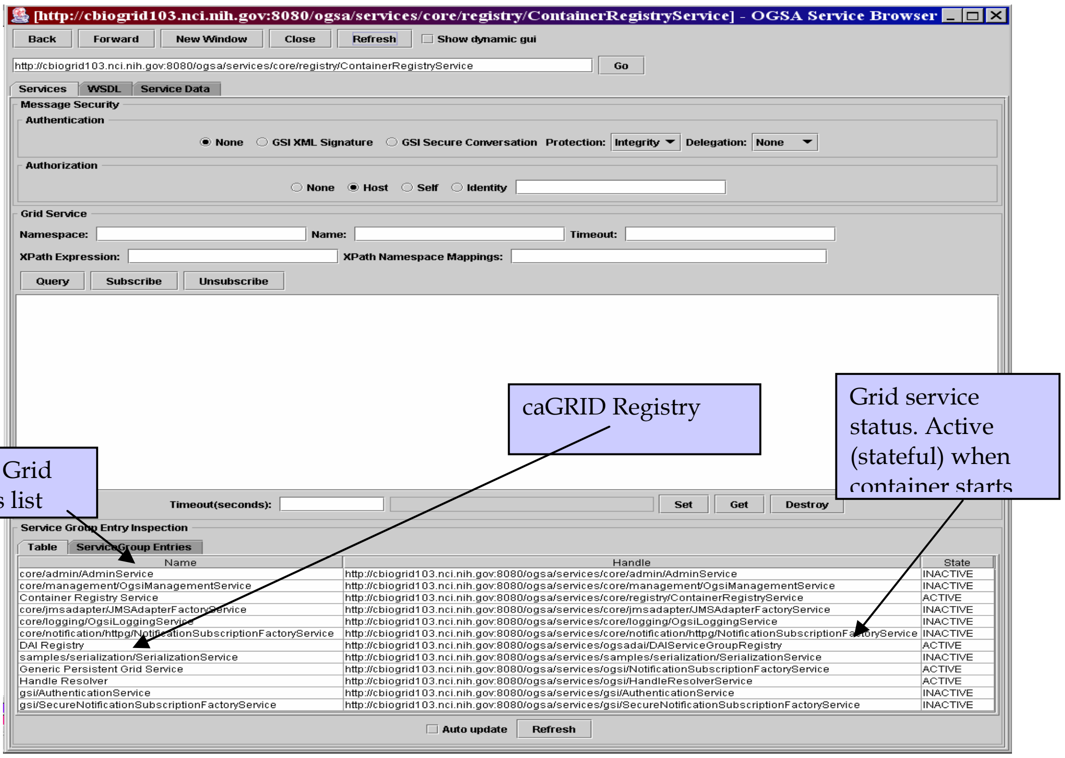
```
[globus@cbiogrid102 bin]$ ./caDiscovery ClinicalTrialProtocol
<?xml version="1.0" encoding="UTF-8"?>
<searchResult>
  <search name="ClinicalTrialProtocol" id="DA512A97-27D5-16D5-E034-0003BA0B1A09">
    <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-Disease" id="DA512A97-57B2-16D5-E034-0003BA0B1A09">
    <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-Agent" id="DA512A97-57BA-16D5-E034-0003BA0B1A09">
    <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-ProtocolAssociation" id="DA512A97-57A9-16D5-E034-0003BA0B1A09">
    <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol" id="CD82C5BE-10E7-349D-E034-0003BA12F5E7">
    <handle url="http://cbiogrid101.nci.nih.gov:8080/ogsa/services/ogsadai/Duke_caBIODataServiceFactory" />
    <handle url="http://156.40.48.168:8080/ogsa/services/ogsadai/GeorgeTown_caBIODataServiceFactory" />
    <handle url="http://jforge.net:8080/ogsa/services/ogsadai/Panther1_caBIODataServiceFactory" />
    <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-PersistentCaBIOBean" id="DA512A97-57C2-16D5-E034-0003BA0B1A09">
    <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-Agent" id="CD82C5BE-1682-349D-E034-0003BA12F5E7">
    <handle url="http://cbiogrid101.nci.nih.gov:8080/ogsa/services/ogsadai/Duke_caBIODataServiceFactory" />
    <handle url="http://156.40.48.168:8080/ogsa/services/ogsadai/GeorgeTown_caBIODataServiceFactory" />
    <handle url="http://jforge.net:8080/ogsa/services/ogsadai/Panther1_caBIODataServiceFactory" />
    <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-ProtocolAssociation" id="CD82C5BE-1683-349D-E034-0003BA12F5E7">
    <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
  <search name="ClinicalTrialProtocol-Agent" id="CD82C5BE-1684-349D-E034-0003BA12F5E7">
    <handle url="http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory" />
  </search>
</searchResult>
```

Figure C-2: Discovery client shows search result output for 'ClinicalTrialProtocols' concept



```
<gridDataServiceResponse xmlns="http://ogsadai.org.uk/namespaces/2003/07/gds/types"><request status="COMPLETED"/><result name="myActivity" status="COMPLETE
"/><result name="statementOutput" status="COMPLETED"><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<nci-core>
    <gov.nih.nci.caBIO.bean.ClinicalTrialProtocol id="70322" xmlns:xlink="http://www.w3.org/1999/xlink/">
        <currentStatus>Active</currentStatus>
        <participationType>Multi-Center</participationType>
        <treatmentFlag>Yes</treatmentFlag>
        <leadOrganizationId>TX098</leadOrganizationId>
        <title>Phase I Study to Determine the Safety, Tolerance and
            Preliminary Antineoplastic Activity of Combined  EGFR
            (erbB1) and HER-2 (erbB2) Blockade, with OSI-774 and
            Trastuzumab, in Combination with Weekly Paclitaxel</title>
        <nihAdminCode>CTEP</nihAdminCode>
        <documentNumber>5439</documentNumber>
        <leadOrganizationName>Cancer Therapy and Research Center</leadOrganizationName>
        <phase>I</phase>
        <currentStatusDate>2002-05-02</currentStatusDate>
        <id>70322</id>
        <piName>Johann S. De Bono</piName>
        <Disease xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=Disease&amp;crit_clinicalTrialProtocols_id=70322"/>
        <Agent xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=Agent&amp;crit_clinicalTrialProtocols_id=70322"/>
        <ProtocolAssociation xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=ProtocolAssociation&amp;crit_clinicalTrialProtocol_id=70322"/>
    </gov.nih.nci.caBIO.bean.ClinicalTrialProtocol>
    <gov.nih.nci.caBIO.bean.ClinicalTrialProtocol id="70654" xmlns:xlink="http://www.w3.org/1999/xlink/">
        <currentStatus>Complete</currentStatus>
        <participationType>Consortium</participationType>
        <treatmentFlag>Yes</treatmentFlag>
        <leadOrganizationId>AMC</leadOrganizationId>
        <title>A Phase I Trial of Combination Bryostatin-1 and
            Vincristine in HIV-Related B-cell Neoplasms</title>
        <nihAdminCode>CTEP</nihAdminCode>
        <documentNumber>AMC-029</documentNumber>
        <leadOrganizationName>AIDS-Associated  Malignancies Clinical
            Trials Consortium</leadOrganizationName>
        <phase>I</phase>
        <pdqIdentifier>AMC-029</pdqIdentifier>
        <currentStatusDate>2003-07-11</currentStatusDate>
        <id>70654</id>
        <piName>Scot C. Remick</piName>
        <Disease xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=Disease&amp;crit_clinicalTrialProtocols_id=70654"/>
        <Agent xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=Agent&amp;crit_clinicalTrialProtocols_id=70654"/>
        <ProtocolAssociation xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=ProtocolAssociation&amp;crit_clinicalTrialProtocol_id=70654"/>
    </gov.nih.nci.caBIO.bean.ClinicalTrialProtocol>
    <gov.nih.nci.caBIO.bean.ClinicalTrialProtocol id="70624" xmlns:xlink="http://www.w3.org/1999/xlink/">
        <currentStatus>Temporarily Closed to Accrual</currentStatus>
        <participationType>Consortium</participationType>
        <treatmentFlag>Yes</treatmentFlag>
        <leadOrganizationId>COGC</leadOrganizationId>
        <title>A Phase I Trial of G3139 (Bcl-2 Antisense, NSC# 683428,
            IND# 58842) Combined with Cytotoxic Chemotherapy in Relapsed
            Childhood Solid Tumors</title>
        <nihAdminCode>CTEP</nihAdminCode>
        <documentNumber>ADVL0211</documentNumber>
        <leadOrganizationName>COG Phase 1 Consortium</leadOrganizationName>
        <phase>I</phase>
        <currentStatusDate>2003-09-10</currentStatusDate>
```

Figure C-3: Query client for caGRID shows search result output in XML format



[http://cbiogrid103.nci.nih.gov:8080/ogsa/services/core/registry/ContainerRegistryService] - OGSA Service Browser

http://cbiogrid103.nci.nih.gov:8080/ogsa/services/core/registry/ContainerRegistryService

Message Security — Authentication: [✓ None] GSI XML Signature, GSI Secure Conversation; Protection: Integrity; Delegation: None

Authorization: None, [✓ Host], Self, Identity

Service Group Entry Inspection

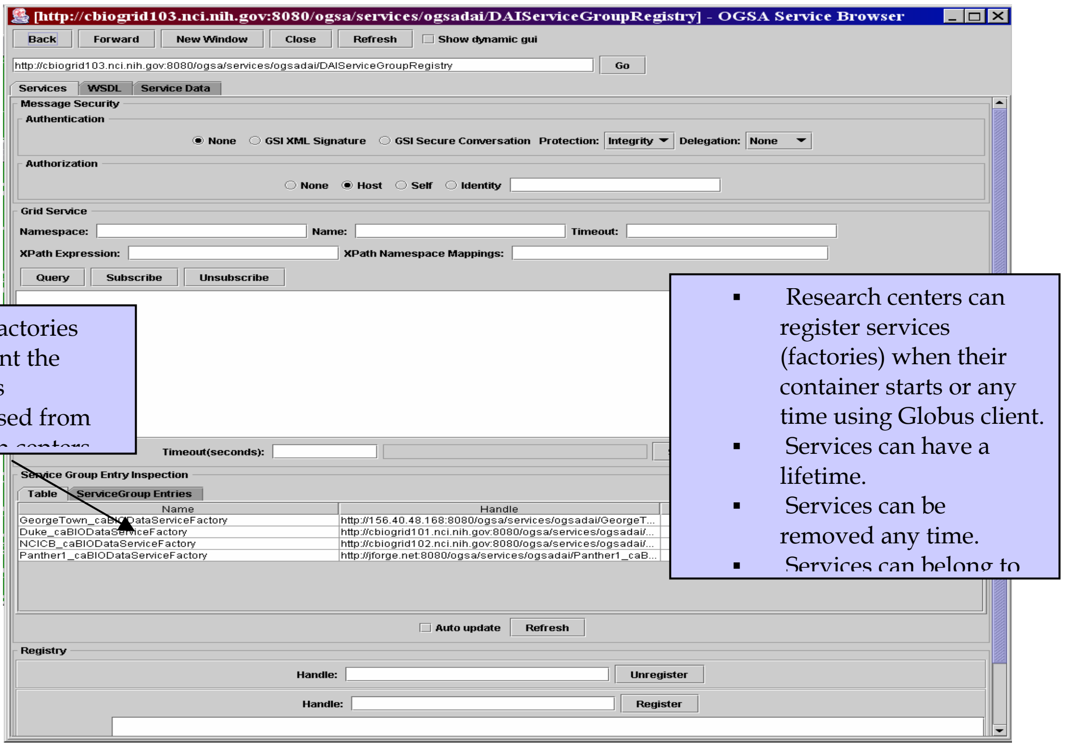
Name	Handle	State
core/admin/AdminService	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/core/admin/AdminService	INACTIVE
core/management/OgsiManagementService	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/core/management/OgsiManagementService	INACTIVE
Container Registry Service	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/core/registry/ContainerRegistryService	ACTIVE
core/jmsadapter/JMSAdapterFactoryService	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/core/jmsadapter/JMSAdapterFactoryService	INACTIVE
core/logging/OgsiLoggingService	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/core/logging/OgsiLoggingService	INACTIVE
core/notification/httpg/NotificationSubscriptionFactoryService	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/core/notification/httpg/NotificationSubscriptionFactoryService	INACTIVE
DAI Registry	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/ogsadai/DAIServiceGroupRegistry	ACTIVE
samples/serialization/SerializationService	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/samples/serialization/SerializationService	INACTIVE
Generic Persistent Grid Service	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/ogsi/NotificationSubscriptionFactoryService	ACTIVE
Handle Resolver	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/ogsi/HandleResolverService	ACTIVE
gsi/AuthenticationService	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/gsi/AuthenticationService	INACTIVE
gsi/SecureNotificationSubscriptionFactoryService	http://cbiogrid103.nci.nih.gov:8080/ogsa/services/gsi/SecureNotificationSubscriptionFactoryService	INACTIVE

caGRID Registry

Grid service status. Active (stateful) when container starts

Globus Grid services list

Figure C-4: Globus client GUI (part of Globus Toolkit) pointing to the NCICB Grid registry server



[http://cbiogrid103.nci.nih.gov:8080/ogsa/services/ogsadai/DAIServiceGroupRegistry] - OGSA Service Browser

http://cbiogrid103.nci.nih.gov:8080/ogsa/services/ogsadai/DAIServiceGroupRegistry

Message Security — Authentication: [✓ None] GSI XML Signature, GSI Secure Conversation; Protection: Integrity; Delegation: None

Authorization: None, [✓ Host], Self, Identity

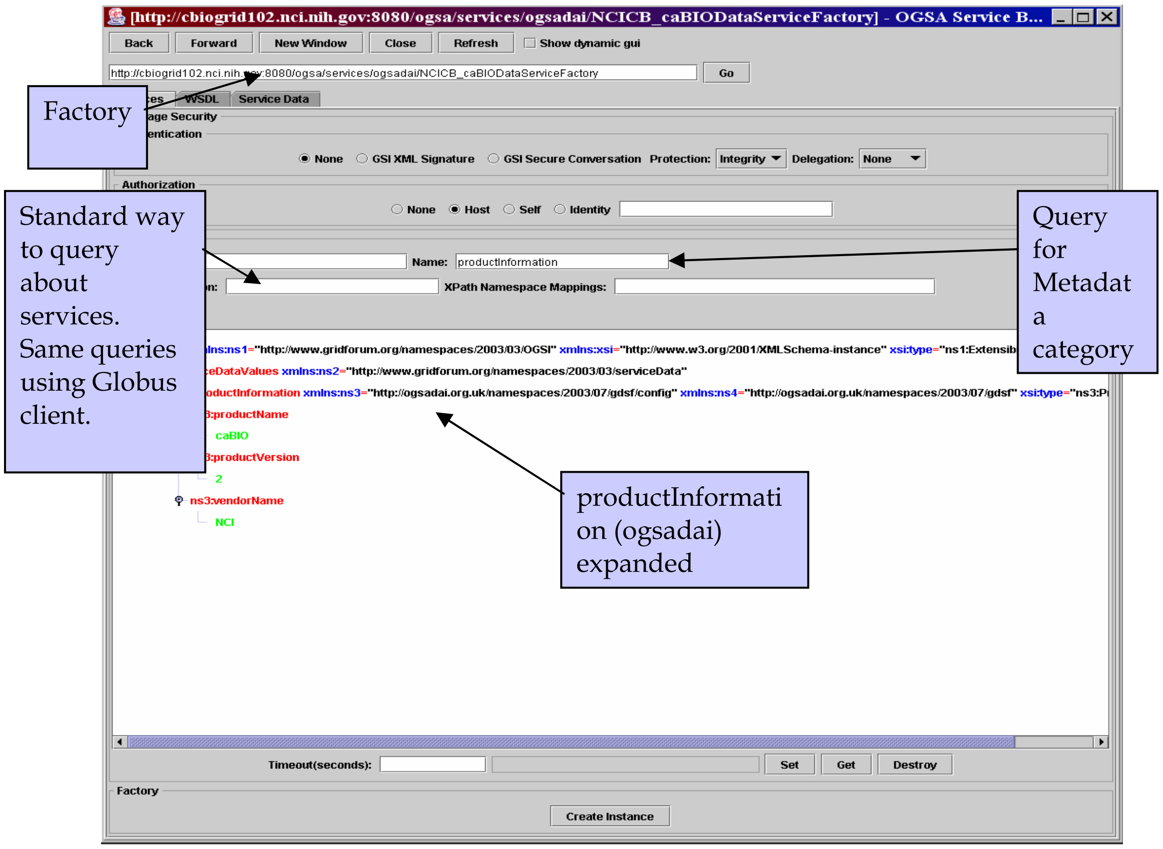
Service Group Entry Inspection

Name	Handle
GeorgeTown_caBIODataServiceFactory	http://156.40.48.168:8080/ogsa/services/ogsadai/GeorgeT...
Duke_caBIODataServiceFactory	http://cbiogrid101.nci.nih.gov:8080/ogsa/services/ogsadai/...
NCICB_caBIODataServiceFactory	http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/...
Panther1_caBIODataServiceFactory	http://jforge.net:8080/ogsa/services/ogsadai/Panther1_caB...

These factories represent the services advertised from research centers

- Research centers can register services (factories) when their container starts or any time using Globus client.
- Services can have a lifetime.
- Services can be removed any time.
- Services can belong to

Figure C-5: Globus client GUI pointing to NCICB OGSA-DAI registry server (Lists all registered Factories)


Factory
Standard way to query about services. Same queries using Globus client.
Query for Metadata category
productInformation (ogsadai) expanded
Back
Forward
New Window
Close
Refresh
Show dynamic gui
Go
WSDL
Service Data
Set
Get
Destroy
Create Instance

[http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory] - OGSA Service B...

http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory

age Security
entication
[✓ None] GSI XML Signature GSI Secure Conversation Protection: Integrity Delegation: None

Authorization
None [✓ Host] Self Identity

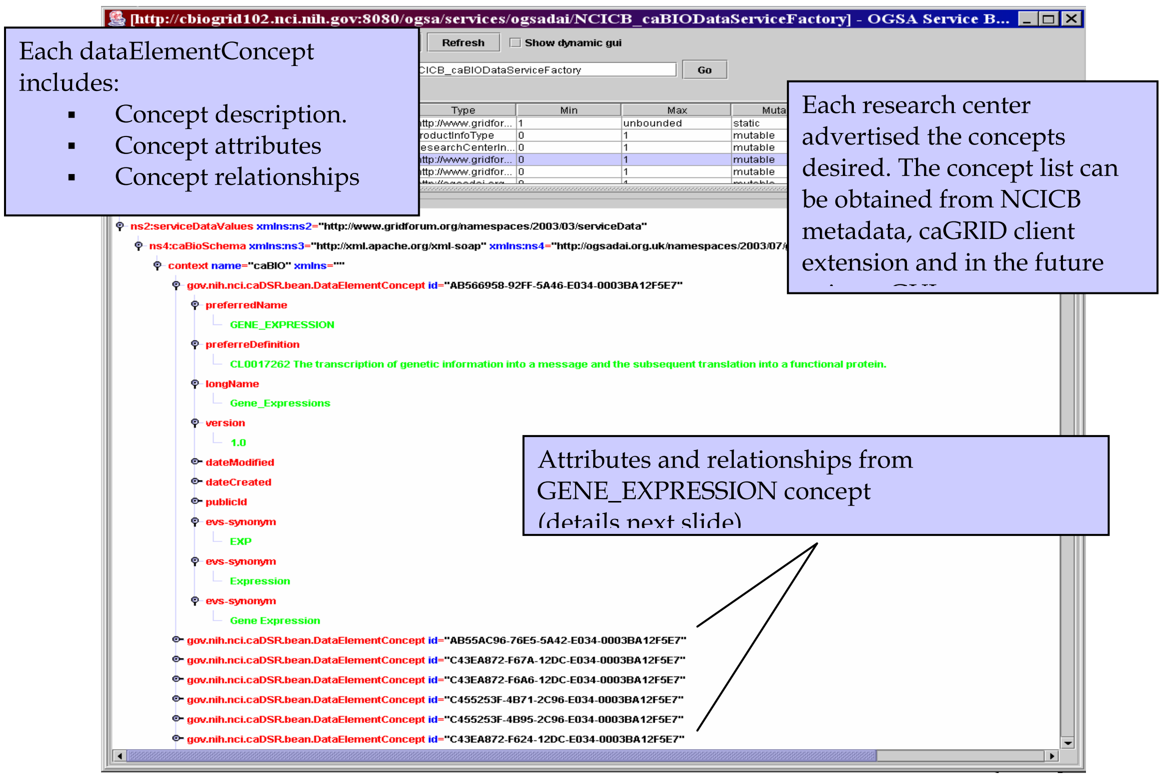
Name: productInformation
n: XPath Namespace Mappings:

```
lns:ns1="http://www.gridforum.org/namespaces/2003/03/OGSI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns1:Extensib
eDataValues xmlns:ns2="http://www.gridforum.org/namespaces/2003/03/serviceData"
oductInformation xmlns:ns3="http://ogsadai.org.uk/namespaces/2003/07/gdsf/config" xmlns:ns4="http://ogsadai.org.uk/namespaces/2003/07/gdsf" xsi:type="ns3:P
3:productName
    caBIO
3:productVersion
    2
ns3:vendorName
    NCI
```

Timeout(seconds):

Factory

Figure C-6: Globus service browser lists Metadata associated with the Factory


Each dataElementConcept includes:
▪ Concept description.
▪ Concept attributes
▪ Concept relationships
Each research center advertised the concepts desired. The concept list can be obtained from NCICB metadata, caGRID client extension and in the future
Attributes and relationships from GENE_EXPRESSION concept (details next slide)
Refresh
Show dynamic gui
Go

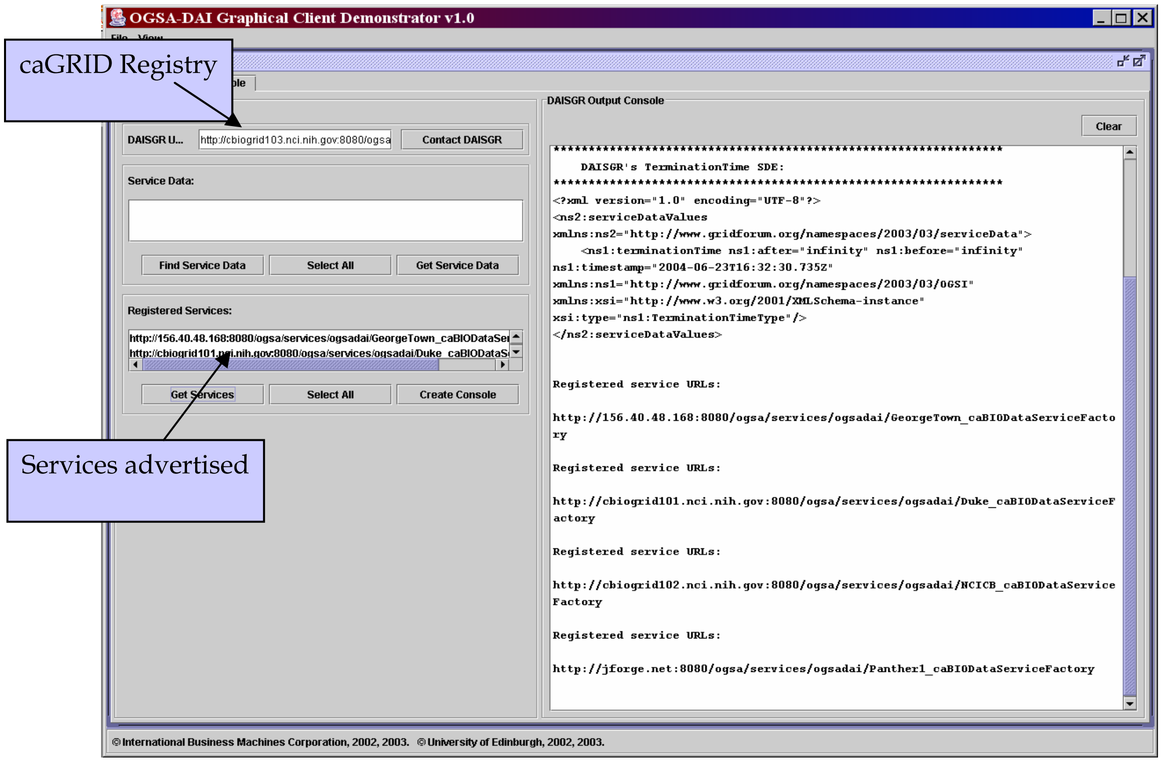
[http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataServiceFactory] - OGSA Service B...

CICB_caBIODataServiceFactory

Type	Min	Max	Muta
ttp://www.gridfor...	1	unbounded	static
roductInfoType	0	1	mutable
esearchCenterIn...	0	1	mutable
ttp://www.gridfor...	0	1	mutable
ttp://www.gridfor...	0	1	mutable

```
ns2:serviceDataValues xmlns:ns2="http://www.gridforum.org/namespaces/2003/03/serviceData"
  ns4:caBioSchema xmlns:ns3="http://xml.apache.org/xml-soap" xmlns:ns4="http://ogsadai.org.uk/namespaces/2003/07/
    context name="caBIO" xmlns=""
      gov.nih.nci.caDSR.bean.DataElementConcept id="AB566958-92FF-5A46-E034-0003BA12F5E7"
        preferredName
          GENE_EXPRESSION
        preferreDefinition
          CL0017262 The transcription of genetic information into a message and the subsequent translation into a functional protein.
        longName
          Gene_Expressions
        version
          1.0
        dateModified
        dateCreated
        publicId
        evs-synonym
          EXP
        evs-synonym
          Expression
        evs-synonym
          Gene Expression
      gov.nih.nci.caDSR.bean.DataElementConcept id="AB55AC96-76E5-5A42-E034-0003BA12F5E7"
      gov.nih.nci.caDSR.bean.DataElementConcept id="C43EA872-F67A-12DC-E034-0003BA12F5E7"
      gov.nih.nci.caDSR.bean.DataElementConcept id="C43EA872-F6A6-12DC-E034-0003BA12F5E7"
      gov.nih.nci.caDSR.bean.DataElementConcept id="C455253F-4B71-2C96-E034-0003BA12F5E7"
      gov.nih.nci.caDSR.bean.DataElementConcept id="C455253F-4B95-2C96-E034-0003BA12F5E7"
      gov.nih.nci.caDSR.bean.DataElementConcept id="C43EA872-F624-12DC-E034-0003BA12F5E7"
```

Figure C-7: Navigating the Service Metadata with Globus GUI



OGSA-DAI Graphical Client Demonstrator v1.0

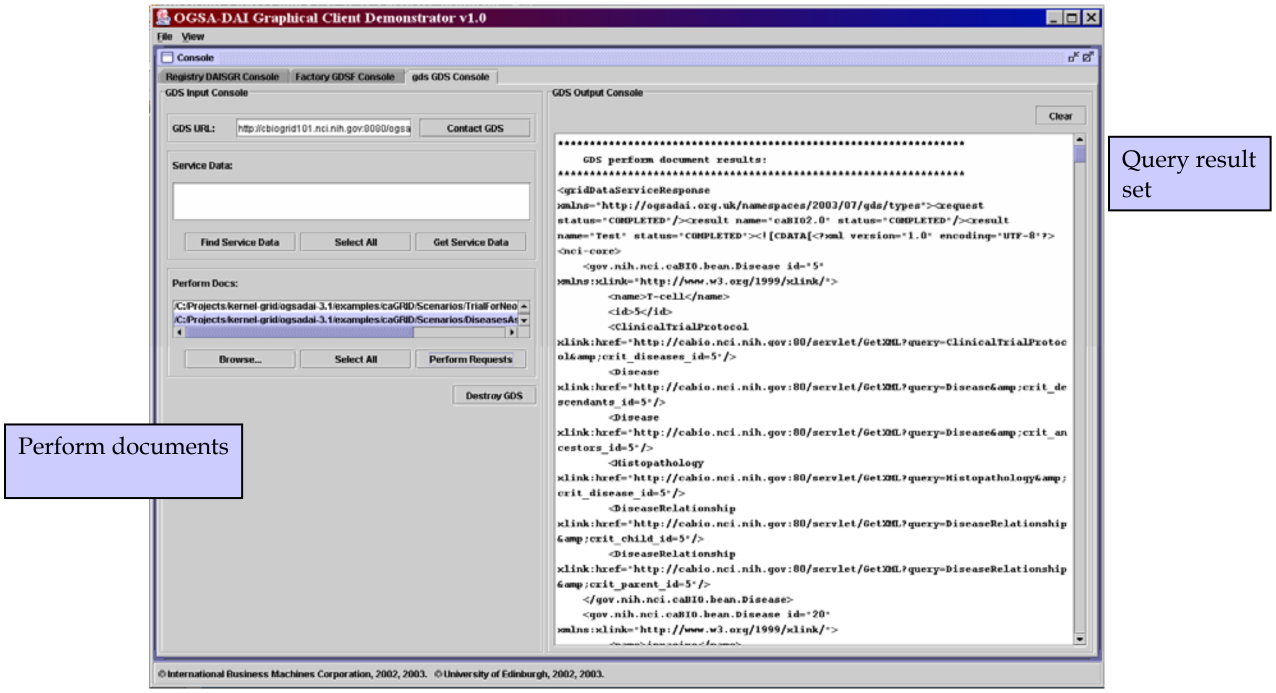
DAISGR U... http://cbiogrid103.nci.nih.gov:8080/ogsa

Registered Services:
http://156.40.48.168:8080/ogsa/services/ogsadai/GeorgeTown_caBIODataSer
http://cbiogrid101.nci.nih.gov:8080/ogsa/services/ogsadai/Duke_caBIODataS

```
*****************************************************************
    DAISGR's TerminationTime SDE:
*****************************************************************
<?xml version="1.0" encoding="UTF-8"?>
<ns2:serviceDataValues
xmlns:ns2="http://www.gridforum.org/namespaces/2003/03/serviceData">
    <ns1:terminationTime ns1:after="infinity" ns1:before="infinity"
ns1:timestamp="2004-06-23T16:32:30.735Z"
xmlns:ns1="http://www.gridforum.org/namespaces/2003/03/OGSI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns1:TerminationTimeType"/>
</ns2:serviceDataValues>


Registered service URLs:

http://156.40.48.168:8080/ogsa/services/ogsadai/GeorgeTown_caBIODataServiceFacto
ry

Registered service URLs:

http://cbiogrid101.nci.nih.gov:8080/ogsa/services/ogsadai/Duke_caBIODataServiceF
actory

Registered service URLs:

http://cbiogrid102.nci.nih.gov:8080/ogsa/services/ogsadai/NCICB_caBIODataService
Factory

Registered service URLs:

http://jforge.net:8080/ogsa/services/ogsadai/Panther1_caBIODataServiceFactory
```

© International Business Machines Corporation, 2002, 2003. © University of Edinburgh, 2002, 2003.

Figure C-8: OGSA-DAI client GUI pointing to the caGRID Registry at NCICB



GDS URL: http://cbiogrid101.nci.nih.gov:8080/ogsa

Perform Docs:
C:\Projects\kernel-grid\ogsadai-3.1\examples\caGRID\Scenarios\TrialForNeo
C:\Projects\kernel-grid\ogsadai-3.1\examples\caGRID\Scenarios\DiseasesAs

```
*******************************************************************
    GDS perform document results:
*******************************************************************
<gridDataServiceResponse
xmlns="http://ogsadai.org.uk/namespaces/2003/07/gds/types"><request
status="COMPLETED"/><result name="caBIO2.0" status="COMPLETED"/><result
name="Test" status="COMPLETED"><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<nci-core>
    <gov.nih.nci.caBIO.bean.Disease id="5"
xmlns:xlink="http://www.w3.org/1999/xlink/">
        <name>T-cell</name>
        <id>5</id>
        <ClinicalTrialProtocol
xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=ClinicalTrialProtoc
ol&amp;crit_diseases_id=5"/>
        <Disease
xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=Disease&amp;crit_de
scendants_id=5"/>
        <Disease
xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=Disease&amp;crit_an
cestors_id=5"/>
        <Histopathology
xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=Histopathology&amp;
crit_disease_id=5"/>
        <DiseaseRelationship
xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=DiseaseRelationship
&amp;crit_child_id=5"/>
        <DiseaseRelationship
xlink:href="http://cabio.nci.nih.gov:80/servlet/GetXML?query=DiseaseRelationship
&amp;crit_parent_id=5"/>
    </gov.nih.nci.caBIO.bean.Disease>
    <gov.nih.nci.caBIO.bean.Disease id="20"
xmlns:xlink="http://www.w3.org/1999/xlink/">
```

©International Business Machines Corporation, 2002, 2003. ©University of Edinburgh, 2002, 2003.

Figure C-9: Submitting a perform document using OGSA-DAI client GUI

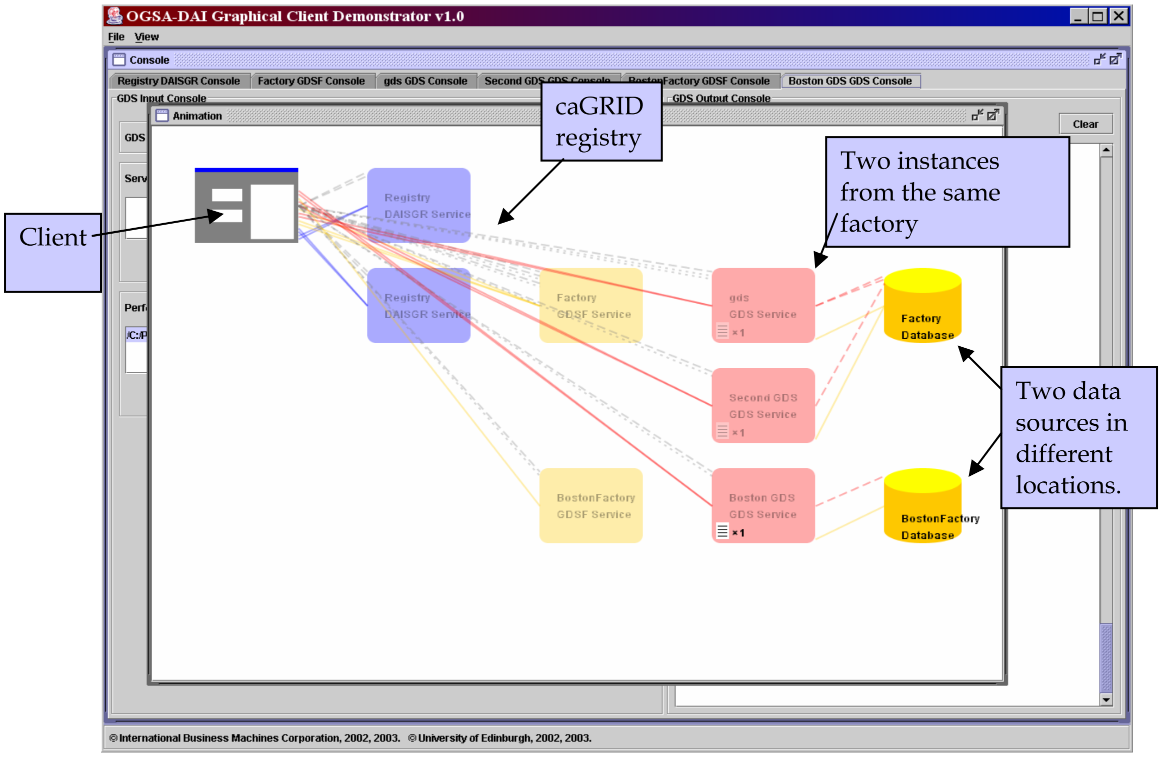


Figure C-10: Animated view of a grid service discovery and query