

IST-2001-32133

GridLab - A Grid Application Toolkit and Testbed

## GridLab And Application Portlets Design

---

Author(s):	Jason Novotny, Michael Russell, Oliver Wehrens
Document Filename:	GridLab-4-1.2-0006-GridLabPortlets
Work package:	Work Package 4: Portals
Partner(s):	MPG
Lead Partner:	MPG
Config ID:	GridLab-4-1.2-0006-1.0
Document classification:	PUBLIC

---

**Abstract:** This document specifies the design and implementation of the GridLab portlets including application portlets that are supported in the GridLab project.





## Contents

<b>1</b>	<b>GridLab and Application Portlets</b>	<b>2</b>
<b>2</b>	<b>Integration with Web Services/OGSA services</b>	<b>2</b>
<b>3</b>	<b>GridLab Services</b>	<b>3</b>
3.1	Grid Services . . . . .	3
3.1.1	Credential Management Service . . . . .	3
3.1.2	Resource Management Services . . . . .	4
3.1.3	Resource Description Language . . . . .	6
3.1.4	Resource Profiles . . . . .	6
3.1.5	Grid Resource Management System . . . . .	7
3.1.6	GRAM Job Management . . . . .	7
3.1.7	GSI SSH Job Submission . . . . .	7
3.2	Data Management . . . . .	7
3.2.1	GridFTP . . . . .	7
3.3	Information Services . . . . .	7
3.3.1	Grid Resource Information Services . . . . .	9
<b>4</b>	<b>GridLab Portlets</b>	<b>9</b>
4.1	Testbed Portlets . . . . .	10
4.2	Security Portlets . . . . .	10
4.3	Data Management Portlets . . . . .	10
4.4	Data Visualization Portlets . . . . .	10
4.5	Resource Management Portlets . . . . .	10
4.6	Grid Information Portlets . . . . .	10
4.7	Application Monitoring Portlets . . . . .	11
4.8	Mobile Service Portlets . . . . .	11
<b>5</b>	<b>Application Portlets</b>	<b>11</b>
5.1	Cactus Portlets . . . . .	11
5.1.1	Cactus Components . . . . .	12
5.1.2	CactusInstallationPortlet . . . . .	13
5.1.3	CactusThornListPortlet . . . . .	13
5.1.4	CactusConfigurationPortlet . . . . .	13
5.1.5	CactusParameterFilePortlet . . . . .	14
5.1.6	CactusSimulationPortlet . . . . .	14
5.1.7	CactusSimluationArchivePortlet . . . . .	14
5.1.8	CactusFileAdvertiserPortlet . . . . .	14
5.2	Triana Portlets . . . . .	14
<b>6</b>	<b>Additional Portlets</b>	<b>15</b>
6.1	Software Development Portlets . . . . .	15
6.1.1	GridCVSPortlet . . . . .	15
6.1.2	GridBuildPortlet . . . . .	15

# 1 GridLab and Application Portlets

## 2 Integration with Web Services/OGSA services

The use of Web Services in distributed computing has been growing rapidly as a means for enabling interoperable application to application communication using a standard set of protocols being defined by the W3C. Specifically, the W3C Web Services Architecture Group defines a Web Service as follows:

”A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.” Web Services are defined by a Web Services Definition Language (WSDL)[29]. The WSDL provides an Interface Definition Language (IDL) that is used to describe the operations (method calls) that a web service may invoke as well as protocol bindings used to transport method invocations. Typically, a WSDL interface known as a portType, is all that is needed to construct a proxy object that can invoke the deployed web service. Just as in CORBA or EJB, the act of invoking method calls on the proxy object actually invokes methods on the (remote) web service using the SOAP protocol.

The Simple Object Access Protocol (SOAP)[30] is used for the invocation of web services and consists of a messaging layer described by XML over a transport protocol, often HTTP, although any protocol may be used e.g. SMTP, FTP, Java Messaging Service (JMS) etc. Other emerging technologies such as the Web Services Invocation Framework (WSIF)[31] may also be used for the stub-less invocation of web services.

Registration of web services for dynamic service discovery can be handled using Universal Description Discovery and Integration (UDDI) registries and the Web Service Inspection Language (WSIL).[33] A WSIL document can contain a list of service descriptions or links to a UDDI repository containing service descriptions. Service requestors can use conventional web mechanisms e.g. HTTP GET to retrieve a WSIL document and discover the services that are provided.

The Open Grid Services Architecture (OGSA) is a collaboration between the Globus development group and IBM to provide an open framework for the development of Grid services based on standards developed within the web services community. Grid services are defined to be web services that support an extended set of interfaces each defined by a portType in WSDL. The following list details the Grid service interfaces taken from the Grid Service Specification[28]:

PortType Name	Description
GridService	encapsulates the root behavior of the component model
HandleMap	mapping from GSH to a GSR (see below)
NotificationSource	allows clients to subscribe to notification messages
NotificationSink	defines a single operation for delivering a notification message to the service instance that implements the operation
Factory	standard operation for creation of Grid service instances
Registry	allows clients to register and unregister registry contents

Currently, the interfaces defining a Grid service are evolving within the OGSA community and may be extended to support concurrency and authorization. Grid services are also concerned with the creation of transient service instances that need to be created dynamically and cleanly shutdown when no longer needed. A serviceType is a Grid service definition containing a set of one or more portTypes supported by the Grid service along with additional versioning information. Dynamically created Grid service instances also have a Grid service handle (GSH)

associated with them that contains a unique name used to refer to that service instance. A GSH is represented as a globally unique URL and can return a Grid service reference. A Grid service reference (GSR) is used to maintain information about a Grid service instance that may change over a service's lifetime and is represented as a WSDL document with extensions to reference a serviceType and expiration information.

The Grid Service Specification (GSS) document specifies how clients create, discover and interact with Grid services and provides a detailed technical description for how the GridLab portal will interact with Grid services developed by other Work Packages. The GSS is currently under review and still in the refinement process. The GSS will be adopted as it becomes more standardized.

Currently, an alpha release of the OGSA software exists in Java and is being tested by early adopters. The codebase leverages the Apache AXIS [34] SOAP and WSDL implementation and uses the Tomcat servlet container as a hosting environment. GSI security is enabled in Axis and Tomcat using the non-open source IAIK SSL libraries provided in the Java CoG kit. We plan on replacing dependencies on IAIK with the Sun Java Secure Socket Extensions (JSSE) libraries and the open source BouncyCastle[35] security libraries.

Under the OGSA framework, a client first creates an instance of a service that is specified as a URL and then gets back a GSH expressed in the form of a URL that is used to invoke the service instance. The samples supplied with the OGSA code base demonstrate simple service factories which are used to create service instances whose methods can be invoked by sample clients. As of this writing, the supplied user's documentation is quite sparse. Hopefully, as OGSA gains favor among the user community and in the GridLab project, more detailed instructions on creating new services and steps for generating clients will follow. In the GridLab portal model, OGSA client services will be part of the GridSphere service library.

## 3 GridLab Services

### 3.1 Grid Services

#### 3.1.1 Credential Management Service

Access to Grid resources in the GridSphere architecture is provided by the Grid Security Infrastructure (GSI)[2]. GSI provides support for mutual authentication using X.509 *proxy* certificates. A proxy certificate provides a temporary certificate/key pair that provides single sign-on access and can be used to access resources on a users' behalf. In order to provide access to Grid enabled resources, GridSphere must have access to a user's valid proxy credential. The Myproxy [20] online credential repository provides a secure repository where users can delegate credentials of a limited lifetime for later retrieval via the portal. The credentials are stored and retrieved using a supplied username and password by a Credential Management Service (CMS). The CMS supports multiple credentials per user. In order to support resources both in the GridLab testbed and other resources, users must be able to select a particular credential to be used for accessing the desired resource. We plan on providing a Portlet that will allow a user to configure which credential should be used for a given set of resources. After this initial configuration step, it should be transparent to a portal user which credential is needed for a particular resource. Credentials are loaded directly into memory and promptly destroyed when a user logs out. In the case that users do have access to their credentials, GridSphere will be able to handle passing uploaded credentials to the CMS.

The Credential Management Service UML diagram in Fig. (§1) lists the capabilities provided. The credentialManagementService is also implemented as a secure portlet service allowing it



Figure 1: Credential Manager Service UML Diagram

to provide capabilities for portal administrators to set the allowed credential subject names (distinguished names) that are accepted by the portal. Portal administrators can also reconfigure the myproxy server settings and the lifetime of accepted proxies. Finally, a mapping between hosts and user credentials is managed using the assignCredential method and is also made persistent.

In addition to the CMS, tools will be provided making it easier for scientists to manage their credentials including delegating credentials to the Myproxy repository. We plan on enhancing the JMyproxy[36] GUI tool for our purposes. We also wish to work with the Myproxy project to add support for managing multiple user credentials.

We wish to work with the Security WP (WP-6) to ensure that the configuration of the portal and Myproxy server meet the requirements of the GridLab testbed. We also plan on becoming early adopters of the Community Authorization Service) as/when it becomes available.

### 3.1.2 Resource Management Services

Resource Management services are used to provide job submission capabilities to portal users. Several technologies for the submission and monitoring of running jobs will be incorporated into the portal services framework. Resource Management services will include support for Globus based job submission and the GridLab Resource Management System (GRMS) developed by WP 9. A preliminary UML object class diagram is presented in Fig. (§2):

The Job Submission Service interface provides methods for submitting, stopping and querying

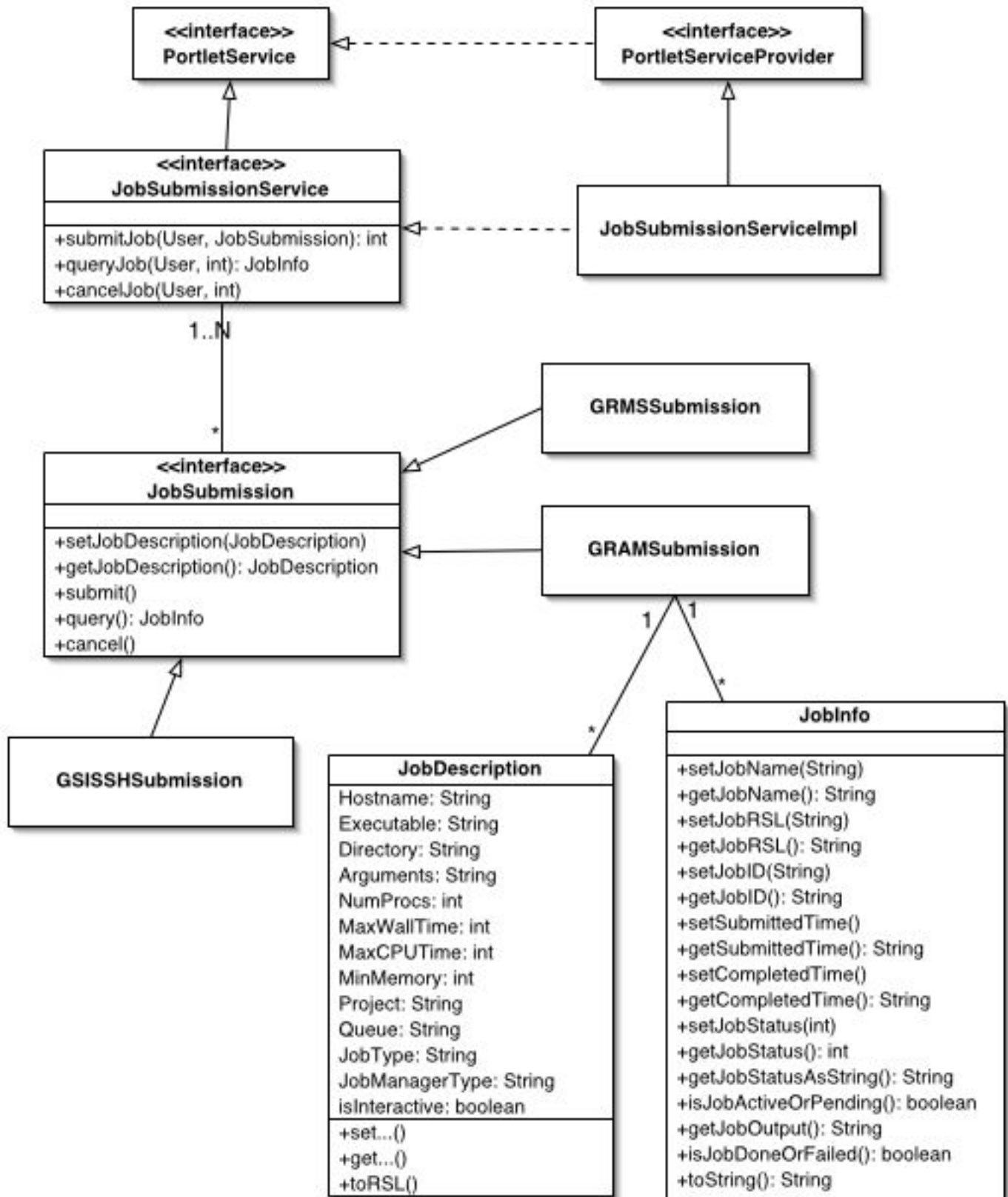


Figure 2: Job Submission Service UML Diagram

jobs. In addition, a background thread can monitor job status periodically and provide the user with email notification when the job completes or allow for the update of job status to a user's profile even after the user logs out.

### 3.1.3 Resource Description Language

Currently, the Globus middleware relies upon the Resource Description Language (RSL) for specifying parameters required for submitting a job. Some of the more useful parameters are listed:

- executable
- arguments
- count (number of processors to use in a parallel job)
- type (can be single or parallel)
- max. wall clock time
- max. CPU time
- max. memory
- min. memory
- queue name

While RSL will continue to be supported for resources supporting Globus 2.0, other services including the GRMS may choose an XML based resource description language. The Java CoG kit also includes prototype support for the conversion of RSL to an XML based format.

### 3.1.4 Resource Profiles

The concept of resource profiles is used to describe the supported job submission services on a particular compute resource. For instance, the portal needs to keep information about which resources have Globus gatekeepers deployed or GSI enabled SSH access, which versions, etc. Wherever possible, we rely upon the Information Services provided by WP 10 to maintain this information and make it available to the portal. However, it may not be feasible to query the GridLab information servers every time a request for a job submission is made. For this reason, the portal will cache information as necessary and provide administrators the ability to update the cache. The list of information necessary regarding job submission is listed:

- Service name – e.g. Globus Gatekeeper or GSI SSH
- Software version
- Port that service is running on
- Names of queues available on resource
- Additional resource information provided by Globus GRIS

### 3.1.5 Grid Resource Management System

The GRMS system developed under WP 9 is a resource broker that attempts to submit a provided job on the best resource available based on any criterion supplied by the user. While GRMS is currently based on Java/Corba technologies, it is the goal of WP 9 to develop a web service or OGSA enabled service for accessing GRMS. Currently, WP 9 has developed a portal page to access GRMS and it will be integrated into a Portlet in the GridSphere portal.

### 3.1.6 GRAM Job Management

Globus Resource and Allocation Management (GRAM)[4] provides an API that uses GSI and is used to submit jobs to Globus gatekeepers. The gatekeeper is responsible for starting up a jobmanager that manages the execution of the job and can handle various scheduling systems including LSF, NQE, LoadLeveler, or PBS to name a few. The goal of the Globus gatekeeper/jobmanager is to hide the differences of local scheduling systems from the user and the portal. The GRAM API also provides mechanisms to cancel jobs and query jobs for status e.g. pending, active, or finished.

### 3.1.7 GSI SSH Job Submission

GSI enabled SSH allows a standard proxy credential to be used to authenticate to a SSH daemon running on a compute resource. OpenSSH has been modified allowing for the more secure version 2 protocol to be used with GSI. Currently, no open source Java implementations are available of OpenSSH, but a version of MindTerm SSH has been augmented to provide support for GSI.

## 3.2 Data Management

Data management services will provide users with the ability to efficiently and easily migrate data between resources (third-party file transfer), as well as the ability to download data or a subset of the data to their client machines. The Data management services will use underlying Grid technology such as GridFTP[5] as well as data management services developed by Data Management and Visualization WP 8.

### 3.2.1 GridFTP

The Grid FTP protocol provides an optimized data transfer library and a specialized set of FTP commands for performing data channel authentication, third-party file transfers, and partial file transfers. GridFTP relies on GSI to perform mutual authentication using X.509 proxy certificates. The Java CoG library provides a Java API for using GridFTP. By using the adaptor pattern to wrap existing CoG GridFTP classes, they can be made more suitable for satisfying the specific needs of GridSphere. The following class diagram, Fig. (§3), illustrates a GridFTPService that will be provided by GridSphere and accessible to Portlets.

## 3.3 Information Services

A critical component of GridSphere is to provide users with information about the GridLab testbed and additional machines that may be external to the testbed. The Information Services work package (WP 10) has proposed the Grid Information Services (GIS) architecture for deploying information services on the GridLab testbed. Information services in GridLab will be used to obtain both static and dynamic information on software and hardware resources.

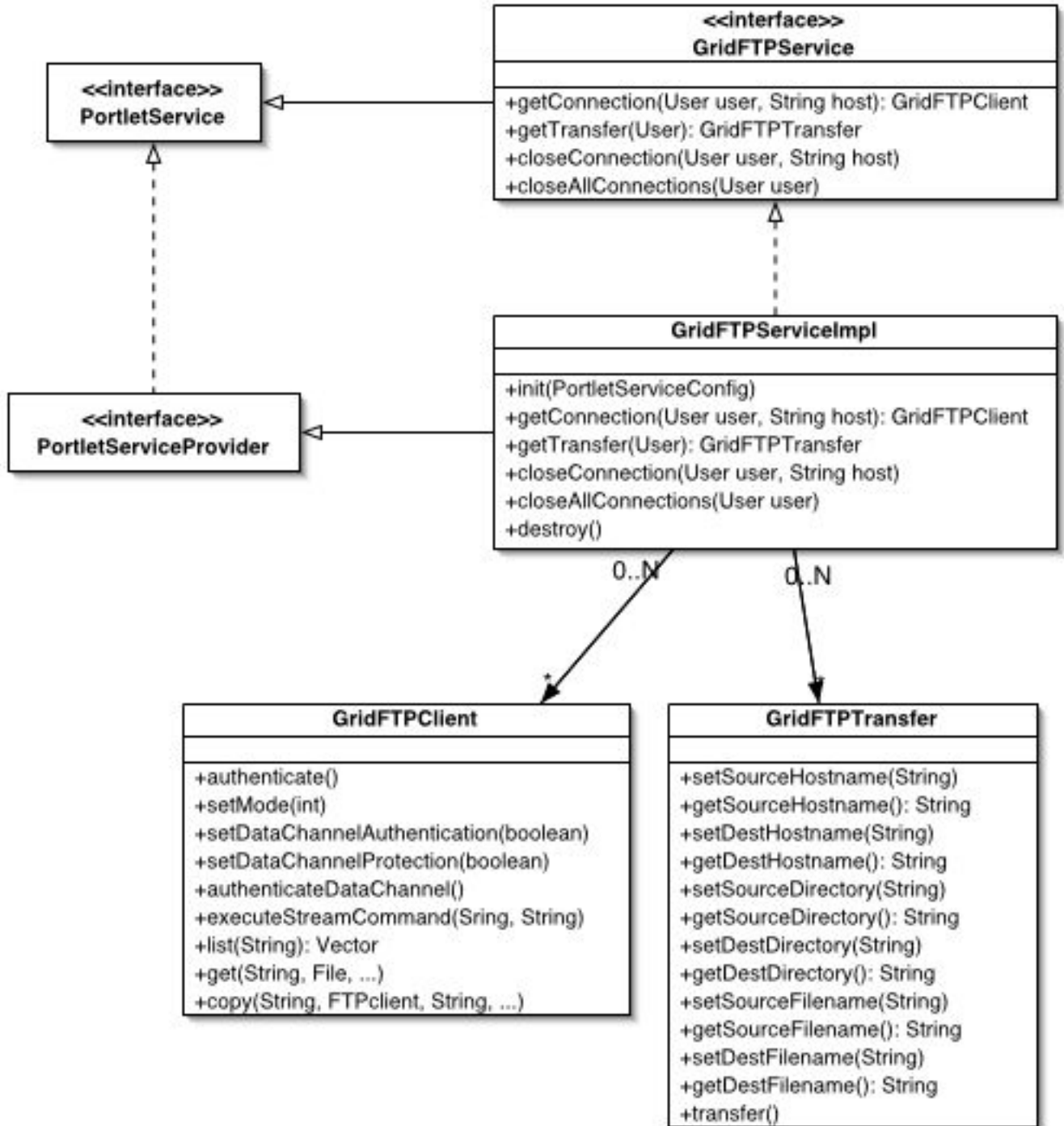


Figure 3: GridFTP Portlet Service

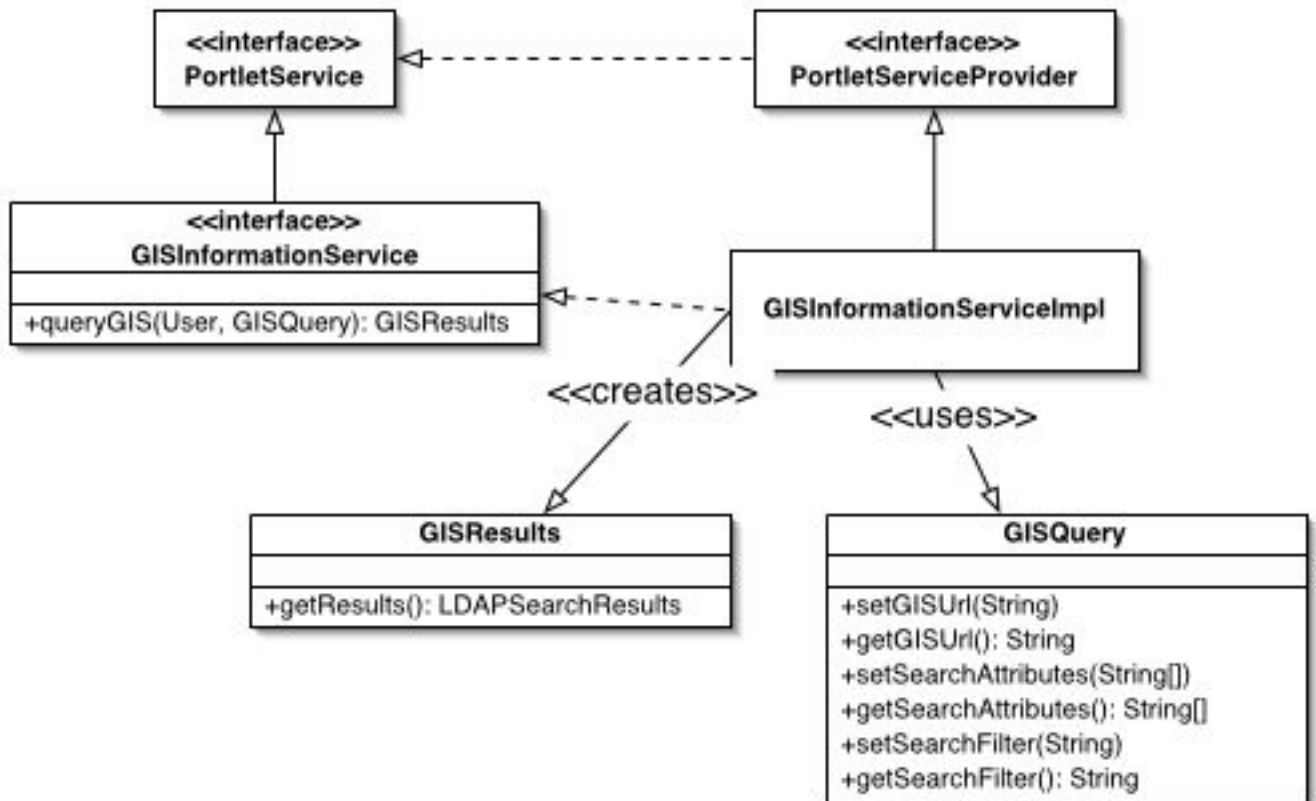


Figure 4: Grid Information Service

### 3.3.1 Grid Resource Information Services

The GIS supports the Lightweight Directory Access Protocol (LDAP) as the communication protocol used to query information services. The Globus toolkit provides an implementation of a Grid Information Service, known as the Grid Information Service (GIS) using OpenLDAP, an open source LDAP server. Also, known as MDS-2, the GIS supports secure authentication based upon GSI and the Simple Authentication and Security Layer (SASL). The latest Java CoG toolkit provides support for the secure MDS-2 using either the Java Naming and Directory Interface (JNDI), or the open source Netscape/Mozilla Directory SDK [44]. An open issue is whether connection pools can be maintained for users to reduce resource consumption, or if users need to re-authenticate for every secure query. Additionally, it may be possible to provide both a secure and non-secure directory tree in the LDAP object class hierarchy.

Fig. (§4), illustrates a first pass at a design of a Grid Information Service to be used by the portal. The GIS service provides a single method queryGIS which takes a GISQuery object describing the query to make and returns a GISResults object in the form of an LDAPSearchResults object if the Netscape SDK is used.

## 4 GridLab Portlets

The Portlets described in the *GridSphere Portlets* section are necessary for supporting a Web portal that links consumers to the core Grid services developed by the community-at-large. However, it is also our goal to assist in the development of Web-interfaces to services developed

by the other GridLab work packages. Because this process is likely to span the entire lifetime of the GridLab project, we mention here only high-level descriptions of the Web-interfaces we expect consumers to require from GridLab services. Furthermore, we concentrate only on those work packages that are developing services to which we can build meaningful Web-interfaces for the consumers of the GridLab Portal. Most notably, work packages that are focused primarily on application library development are not considered here, including the *GAT*, *Cactus-GAT*, *Triana-GAT*, and the *Adaptive Components* work packages. Though, at some future time consumers may demand Web-interfaces that provide value-added services to products developed by these work packages. On the other hand, please see the *Application Portlets* sections for descriptions of Web interfaces we are building for users of Cactus and Triana applications.

#### **4.1 Testbed Portlets**

Testbed Portlets will provide Web-interfaces to libraries and services developed by the Testbed Work Package. The primary consumers of these Portlets will be Grid Testbed administrators, though some Portlets are likely to be useful to general users of the GridLab Portal for discovering and testing specific elements of the GridLab Testbed or other Grid testbeds.

#### **4.2 Security Portlets**

Security Portlets will provide Web-interfaces to libraries and services developed by the Security Work Package. The primary consumers of these Portlets will be administrators of the GridLab Authorization Service for assigning privileges to consumers of GridLab resources and services, including elements of the GridLab Portal.

#### **4.3 Data Management Portlets**

Data Management Portlets will provide Web-interfaces to data management libraries and services developed by the Data Handling and Visualization Work Package. The primary consumers of these Portlets will be scientists that require Web-interfaces for locating, transferring, replicating, and archiving datasets.

#### **4.4 Data Visualization Portlets**

Data Visualization Portlets will provide Web-interfaces to data visualization libraries and services developed by the Data Handling and Visualization Work Package. The primary consumers of these Portlets will be scientists that require Web-interfaces for visualizing and/or launching applications or services to visualize datasets

#### **4.5 Resource Management Portlets**

Resource Management Portlets will provide Web-interfaces to resource management libraries and services developed by the Resource Management Work Package. The primary consumers of these Portlets will include scientists that require Web-interfaces for launching applications on Grid resources and administrators of the GridLab Resource Management Services.

#### **4.6 Grid Information Portlets**

Grid Information Portlets will provide Web-interfaces to libraries and services developed by the Information Services Work Package. The primary consumers of these Portlets will include

GridLab administrators and developers interested in viewing information provided by GridLab Information Services.

#### 4.7 Application Monitoring Portlets

Application Monitoring Portlets will provide Web-interfaces to libraries and services developed by the Application Monitoring Work Package. The primary consumers of these Portlets will include GridLab scientists interested in monitoring closely the performance of their applications and administrators of GridLab Application Monitoring Services.

#### 4.8 Mobile Service Portlets

Mobile Service Portlets will provide Web-interfaces to valued-added services developed by the Mobile Services Work Package.

### 5 Application Portlets

Another goal of our work package is to design and build Portlets for supporting specific applications in the GridLab Portal. The two primary applications we are required to support are *Cactus* and *Triana*, though we are actively searching for other target applications.

#### 5.1 Cactus Portlets

We held extensive meetings with the Cactus Project and users of the Cactus Computational Toolkit at Max-Planck-Institute for Gravitational Physics to identify Cactus user requirements for the GridLab Portal. In particular, we identified the following use-case scenario as the most critical Cactus use-case for the GridLab Portal.

*A numerical relativist at Max-Planck-Institute for Gravitational Physics wants to run a Cactus physics simulation represented by a list of Cactus a parameter file and, optionally, a list of thorns. (Note that the required thorns can be determined from the parameter file.) This physicist would like to be able to instruct the GridLab Portal to locate and reserve the best available computing resource for their simulation, construct a Cactus simulation executable containing the given list of thorns and submit that simulation executable along with the given parameter file to the resource job scheduling system. This physicist would like to be notified, say via email, when the Cactus simulation begins and ends, as well as to be notified at certain time-steps during the course of the simulation. After the simulation ends, they would like the GridLab Portal to archive the data produced by the simulation as well as to make this data available for analysis upon request. Generally, a physicist is interested in visualizing 1-dimensional data while the Cactus simulation is running and immediately after the Cactus simulation ends. It is generally preferable to archive 2- and 3-dimensional datasets, due to their large size, until they are needed for analysis.*

We identified several other requirements that either support this use-case scenario, such as the ability to test new code before committing changes to CVS, or extend this use-case scenario, for example physicists at MPI regularly perform *parameter studies*. Because we require more interaction with the Cactus user base, in the rest of this section we describe only those Portlets that we determined are necessary to support the above use-case scenario and the requirements most essential to the users of Cactus software. Note that in all cases we presume the need to develop portlet services.

### 5.1.1 Cactus Components

#### CactusInstallation

A CactusInstallation is a directory, specified by a unique path, that contains Cactus software. A CactusInstallation is usually generated by calling the GetCactus script, available on the Cactus Project Website, which retrieves files from CVS. The directory structure is as follows:

- arrangements : contains Cactus arrangements and thorns.
- configs : contains Cactus configurations.
- doc : contains documentation.
- lib : contains external libraries necessary to build Cactus.
- src: contains source code for Cactus flesh.

#### CactusFlesh

The CactusFlesh provides the core infrastructure for the Cactus framework.

#### CactusThorn

A CactusThorn is software component in Cactus. All user-supplied code goes into CactusThorns, which are, by and large, independent of each other. CactusThorns communicate via calls to the CactusFlesh API, and more rarely, custom APIs of other CactusThorns.

#### CactusArrangement

CactusThorns are grouped into CactusArrangements. This logical grouping of thorns is purely for organizational purposes. CactusArrangements are located in the arrangements directory.

#### CactusImplementation

A CactusImplementation defines a group of variables and parameters which are used to implement some functionality. Each CactusThorn can provide one or more CactusImplementations and relationships between CactusThorns are defined through these CactusImplementations. For example, the thorn CactusPUGH/PUGH provides the CactusImplementation *driver*. This CactusImplementation is responsible for providing memory for grid variables and for communication.

#### CactusThornList

A CactusThornList is a file that names a list of CactusThorns with the syntax:

```
<arrangement name>/<thorn name>[<cvs repository>]
```

The GetCactus script uses CactusThornLists to retrieve CactusThorns from CVS. By default, each CactusThorn is presumed to be located in the *standard* CactusRepository.

#### CactusRepository

The Cactus Project maintains at least two Cactus software repositories on cvs.cactuscode.org, *standard* and *development*.

### **CactusConfiguration**

A CactusConfiguration is a directory, specified by a unique path and usually located within a CactusInstallation, that contains files for building a CactusExecutable with *autoconf* and *gmake*.

### **CactusExecutable**

A CactusExecutable is a file that is executable on one or more architectures. CactusExecutables are generated by CactusConfigurations.

### **CactusParameterFile**

A CactusParameterFile is a file that provides parameters to a CactusExecutable. CactusParameterFiles contains a list of thorns to activate and usually parameters to those thorns. CactusParameterFile contains the appropriate information for constructing a CactusConfiguration, and hence for generating a CactusExecutable.

### **CactusSimulation**

A CactusSimulation is defined by a CactusParameterFile and one or more processes, where each process represents a CactusExecutable executed with the given CactusParameterFile.

## **5.1.2 CactusInstallationPortlet**

The CactusInstallationPortlet enables users to install and update Cactus software on target hosts with the *getcactus* script. This script is developed and maintained by the Cactus Project and uses cvs checkout and update tools to obtain the Cactus flesh and thorns. It is included in the Cactus CVS standard repository at [cvs.cactuscode.org](http://cvs.cactuscode.org). The *getcactus* script enables users to select from which Cactus repository to obtain the Cactus flesh. It also enables users to provide a CactusThornList. By default, the *getcactus* obtains the latest revision of the flesh and thorns, but it users can also specify which revisions to obtain. The CactusInstallationPortlet will support the options offered by the *getcactus* script.

## **5.1.3 CactusThornListPortlet**

The CactusThornListPortlet enables users to maintain one or more CactusThornLists for use with CactusInstallationPortlet.

## **5.1.4 CactusConfigurationPortlet**

The CactusConfigurationPortlet enables users to create and update Cactus configurations with the *configure* script that is provided when one installs Cactus software with the *getcactus* script. The *configure* script is based on *autoconf*. It produces a directory within a particular Cactus installation and generates a *makefile* and the necessary files to produce a Cactus executable with *gmake*. Thus, *CactusConfigurationPortlet* also enables users to build executables for each Cactus configuration with *gmake*, allowing users to specify appropriate tags and command-line options.

### 5.1.5 CactusParameterFilePortlet

The `CactusParameterFilePortlet` enables users to upload Cactus parameter file to a target resource. When the parameter file is uploaded, users can specify whether they would like to have that parameter file modified to contain parameters that are known to be required for the target resource in order for Cactus simulations using that parameter file to behave correctly. Users may also edit Cactus parameter files online or download Cactus parameter files with this Portlet.

### 5.1.6 CactusSimulationPortlet

The `CactusSimulationPortlet` enables users to run Cactus simulations with the `qs2` script. The `qs2` script is a batch-script written and maintained by the Cactus Project for submitting a Cactus executable and parameter file to batch queues on target resources. The `CactusSimulationPortlet` requires the *minimum information* for running a Cactus simulation:

- *Resource requirements:* Currently, users are very familiar with the resources they employ to run Cactus simulation. The information they are most interested in obtaining from the GridLab Portal is queue status information.
- *Parameter file:* In this case, the `CactusSimulationPortlet` will check to see whether Cactus is installed on the target resource and whether an appropriate configuration exists. If not, it will produce a Cactus configuration given the Cactus parameter file, using the `makethornlist`, `getcactus`, and `configure` scripts provided with Cactus software. And if required, the `CactusSimulationPortlet` will subsequently use `gmake` to build an executable before running the simulation as described above.

The `CactusSimulationPortlet` also enables users to specify whether to produce an archive of a simulation after it has completed. By archiving, we mean that one or more tarballs is created containing a *Cactus executable*, *Cactus parameter file* and *Cactus datasets*. In some cases, users may wish to archive 2- and 3-dimensional data in separate tarballs, due to their large size. This archive information stored on behalf of users is accessible from both the `CactusSimulationPortlet` and the `CactusSimulationArchivePortlet`.

### 5.1.7 CactusSimulationArchivePortlet

The `CactusSimulationArchivePortlet` enables users to tarball and untarball Cactus simulation archives. It also enables users to transfer archives to other locations with GridFTP.

### 5.1.8 CactusFileAdvertiserPortlet

The `CactusSimulationArchivePortlet` enables users to view the files produce during a Cactus simulation and after it completes. It uses the *CactusFileAdvertiser* to make this information available.

## 5.2 Triana Portlets

We are still in the early requirements analysis stages for Triana.

## 6 Additional Portlets

There are a variety of others Portlets we are interested in developing to support other activities not directly represented by work packages in GridLab. In some cases, these Portlets will be developed for research interests, though in all cases we want to provide useful functionality to GridLab consumers.

### 6.1 Software Development Portlets

In this section, we describe Portlets and services that we are interested in developing to support general application development needs.

#### 6.1.1 GridCVSPortlet

The GridCVSPortlet enables users to perform third party gridcvs operations. For example, users would be able to perform *cvs checkout* or *cvs update* to install or update software on target hosts. Here a user must be able to authenticate to the *gridcvs* repository servers they wish to use and to either *gsi-ssh* or *globus-gatekeeper* servers on the target hosts. The GridCVSPortlet also enables users to manage a personal list of *gridcvs* repositories they would like to be made available each time they access this Portlet. Finally, it allows users to test their proxy credentials against those repositories.

#### 6.1.2 GridBuildPortlet

The GridBuildPortlet enables users to run *make* or *gmake* and related commands, such *autoconf* or *libtool* commands on target hosts with either *globus-gatekeeper* or *gsi-ssh* services installed. In order to use this command, users must be able to authenticate to either of these services.

## References

- [1] Gamma E, Helm R, Johnson R, Vlissides J. *Design Patterns: Elements of Reusable Object Oriented Software*. Addison-Wesley, 1995.
- [2] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids. *Proc. 5th ACM Conference on Computer and Communications Security Conference* pp. 83-92, 1998.
- [3] Czajkowski K, Fitzgerald S, Foster I, Kesselman C. Grid Information Services for Distributed Resource Sharing. *Proc. 10th IEEE Symp. On High Performance Distributed Computing* 2001;
- [4] Czajkowski K, Foster I, Karonis N, Kesselman C, Martin S, Smith W, Tuecke S. A Resource Management Architecture for Metacomputing Systems. *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing* 1998;
- [5] Allcock W, Bester J, Bresnahan J, Chervenak A, Liming L, Tuecke S. *Grid Forum Working Draft* March 2001; <http://www.gridforum.org>
- [6] Novotny J. The Grid Portal Development Kit *IEEE Concurrency: Practice and Experience* 2002;

- [7] Apache Webservice Project  
<http://www.apache.org>
- [8] ModSSL  
<http://www.modssl.org>
- [9] Java Servlet 2.3 and Java Server Pages 1.2 Specifications  
<http://java.sun.com/products/servlets>
- [10] The Jakarta Tomcat Project  
<http://jakarta.apache.org/tomcat>
- [11] The Jakarta Jetspeed Project  
<http://jakarta.apache.org/jetspeed>
- [12] The Jakarta Turbine Project  
<http://jakarta.apache.org/turbine>
- [13] The Jakarta ANT Project  
<http://jakarta.apache.org/ant>
- [14] The Jakarta Log4J Project  
<http://jakarta.apache.org/log4j>
- [15] The Jakarta JMeter Project  
<http://jakarta.apache.org/jmeter>
- [16] The JUnit Project  
<http://www.junit.org>
- [17] The Castor Project  
<http://castor.exolab.org/>
- [18] Role Based Access Control links  
<http://csrc.nist.gov/rbac/>
- [19] Laszewski G., Foster I, Gawor J. CoG Kits: A Bridge Between Commodity Distributed Computing and High-Performance Grids *Proceedings of the ACM Java Grande Conference 2000*;
- [20] Novotny J., Tuecke S., Welch V. An Online Credential Repository for the Grid: MyProxy. *Proc. 10th IEEE Symp. On High Performance Distributed Computing 2001*;
- [21] Allen G, Daues G, Foster I, Laszewski G, Novotny J, Russell M, Seidel E, Shalf J. The Astrophysics Simulation Collaboratory Portal: A Science Portal Enabling Community Software Development. *Proc. of the 10th IEEE Intl. Symp. on High Perf. Dist. Comp 2001*;
- [22] What is a Portlet?  
<http://www-3.ibm.com/software/webservers/portal/portlet.html>
- [23] JSR 168: Portlet Specification  
<http://www.jcp.org/jsr/detail/168.jsp>
- [24] Hesmer S., Fischer P., Buckner T., Schuster I. *Portlet Development Guide* April 2, 2002;

- [25] Hesmer S., Hepper S., Schaeck T. *Portlet API (First Draft)* April 2, 2002;
- [26] Kelley, I., Novotny, J., Russell, M., Wehrens, O. *Jetspeed Evaluation WP4 Internal Document* May, 2002;
- [27] I. Foster, C. Kesselman, J. Nick, S. Tuecke *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*. January, 2002.
- [28] Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C. and Nick, J. *Grid Service Specification*
- [29] *Web Services Description Language*  
<http://www.w3.org/TR/wsdl>
- [30] *Simple Object Access Protocol*  
<http://www.w3.org/TR/soap12-part1/>
- [31] *Web Services Invocation Framework*  
<http://www.alphaworks.ibm.com/tech/wsif>
- [32] *Universal Description Discovery and Integration*  
<http://www.uddi.org>
- [33] *Web Service Invocation Language*  
<http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html>
- [34] *Apache Axis*  
<http://xml.apache.org/axis>
- [35] *Legion of the Bouncy Castle*  
<http://www.bouncycastle.org>
- [36] *JMyproxy*  
<ftp://george.lbl.gov/pub/globus/jmyproxy.tar.gz>
- [37] *Model View Controller*  
<http://>
- [38] *Apache Xalan-J*  
<http://xml.apache.org/xalan-j/index.html>
- [39] *Jakarta JDO - OJB*  
<http://jakarta.apache.org/ojb/>
- [40] *Castor O/R Mapper*  
<http://castor.exolab.org>
- [41] *Java Server Pages Standard Tag Library*  
<http://java.sun.com/products/jsp/jstl/>
- [42] *Sun JDO Specification*  
<http://access1.sun.com/jdo/>
- [43] *OpenSymphony JSP Caching Tag Library*  
<http://sourceforge.net/projects/opensymphony/>



- [44] Netscape Directory and LDAP Developer Central.  
<http://developer.netscape.com/tech/directory/index.html>

#### Software Dependencies

In keeping with the requirements, all software used in implementing the GridLab Portal is open-source and preferably under a BSD-style license rather than the more restrictive GNU Public License (GPL).

- Jakarta Log4J Logging Library[14]
- JUnit Testing Library[16]
- OpenSymphony JSP Caching Tag Library[43]
- Java Server Pages Standard Tag Library[41]
- Bouncy Castle Security Libraries[35]