GWD-I (draft-ggf-ogsa-spec-016)
Open Grid Services Architecture
http://forge.gridforum.org/projects/ogsa-wg

Editors:
I. Foster, Argonne & U.Chicago
H. Kishimoto, Fujitsu

May 17, 2004

The Open Grid Services Architecture, Version 1.0

Status of this Memo

This document provides information to the community regarding the specification of the Open Grid Services Architecture (OGSA). Distribution of this document is unlimited. This is a DRAFT document and continues to be revised.

This document is being constantly updated. The latest version can be found at:
https://forge.gridforum.org/projects/ogsa-wg/document/draft-ggf-ogsa-spec/en/

Abstract

Successful realization of the Open Grid Services Architecture (OGSA) vision of a broadly applicable and adopted framework for distributed system integration, virtualization, and management requires the definition of a core set of interfaces, behaviors, resource models, and bindings. This document, produced by the OGSA working group within the Global Grid Forum (GGF), provides a first, and necessarily preliminary and incomplete, version of this OGSA definition. The document specifies the scope of important services required to support Grid systems and applications in both e-science and e-business, identifies a core set of such services that are viewed as essential for many systems and applications, and specifies at a high-level the functionalities required for these core services and the interrelationships among those core services. The document also lists existing technical standards and standard definition activities within GGF, OASIS, W3C, and other standards bodies that speak to required OGSA functionality, and identifies priority areas for further work.

ogsa-wg@ggf.org

GLOBAL GRID FORUM
office@ggf.org
www.ggf.org

Full Copyright Notice

Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director (see contact information at GGF website).

Contents

1 Introduction

Grid systems and applications aim to integrate, virtualize, and manage resources and services within distributed, heterogeneous, dynamic "virtual organizations" [6, 7]. The realization of this goal requires the disintegration of the numerous barriers that normally separate different computing systems within and across organizations, so that computers, application services, data, and other resources can be accessed as and when required, regardless of physical location.

Key to the realization of this Grid vision is standardization, so that the diverse components that make up a modern computing environment can be discovered, accessed, allocated, monitored, accounted for, billed for, etc., and in general managed as a single virtual system—even when provided by different vendors and/or operated by different organizations. Standardization is critical if we are to create interoperable, portable, and reusable components and systems; it can also contribute to the development of secure, robust, and scalable Grid systems by facilitating the use of good practices.

In this document, we present an *Open Grid Services Architecture* (OGSA) that addresses this need for standardization by defining, within a service-oriented architecture, a set of core interfaces and behaviors that address key concerns in Grid systems. These concerns include such issues as: How do I establish identity and negotiate authentication? How is policy expressed and negotiated? How do I discover services? How do I negotiate and monitor service level agreements? How do I manage membership of, and communication within, virtual organizations? How do I organize service collections hierarchically so as to deliver reliable and scalable service semantics? How do I integrate data resources into computations? How do I monitor and manage collections of services?

In our presentation, we first identify required capabilities for Grid systems and then translate those capabilities into a coherent set of components that collectively define OGSA. First, in §2, we provide an abstract definition of the set of requirements that OGSA is intended to address. This analysis is based on requirements, technical challenges, previous experience, and the state of the art in related work. The abstract rendering is not constrained by any assumptions about the underlying infrastructure, but rather is intended to frame the "Grid" discussion and define solutions.

In §3, we present a rendering of the required functionality into service definitions. In addition, in §3.8, we describe infrastructure assumptions that constrain our development of the OGSA design. In particular, we explain how OGSA both builds on, and is contributing to the development of, the growing collection of technical specifications that form the emerging Web services (WS) architecture [2].

In a later draft we will provide descriptions of specific services, making clear where existing service specifications can be used unchanged and where modified or new service specifications are needed. We will also describe the current state of any work underway to define such extensions and/or definitions.

This document, a product of the Global Grid Forum's OGSA working group, defines OGSA version 1. The OGSA working group is likely to release revisions of this document in the future as our understanding of OGSA's purpose and form, and the details of specific components, evolves.

2 Requirements

Only slightly updated according to May'04 F2F discussions.

This definition of OGSA version 1 is driven by a set of functional requirements, which themselves are informed by the use cases listed in Table 1 and described in a companion document [4]. These use cases do not constitute a formal requirements analysis, but have provided useful input to the definition process.

Table 1 The OGSA use cases

Use case	Summary
Commercial Data Center (CDC)	Data centers will have to manage several thousands of IT resources, including servers, storage, and networks, while reducing management costs and increasing resource utilization.
Severe Storm Modeling	Enable accurate prediction of the exact location of severe storms based on a combination of real-time wide area weather instrumentation and large-scale simulation coupled with data modeling.
Online Media and Entertainment	Delivering an entertainment experience, either for consumption or interaction.
National Fusion Collaboratory (NFC)	Defines a virtual organization devoted to fusion research and addresses the need of software developed and executed by this community based on the application service provider (ASP) model.
Service-Based Distributed Query Processing	A service-based distributed query processor supporting the evaluation of queries expressed in a declarative language over one or more existing services.
Grid Workflow	Workflow is a convenient way of constructing new services by composing existing services. A new service can be created and used by registering a workflow definition to a workflow engine.
Grid Resource Reseller	Inserting a supply chain between the Grid resource owners and end users will allow the resource owner to concentrate on their core competence, while end users can purchase resources bundled into attractive packages by the reseller.
Inter Grid	Extends the CDC use case by emphasizing plethora of applications that are not grid enabled and are difficult to change, mixed grid and non grid data centers, grid across multiple companies, etc. Also brings into view generic concepts of utility computing.
Interactive Grids	Compared to the online media use case, this use case emphasizes a very high granularity of distributed execution.
Grid Lite	Extends the use of grids to small devices – PDAs, cell phones, firewalls etc, and identifies a set of essential grid services that enable the device to be part of a grid environment.
Virtual Organization Grid Portal	A VO gives its members access to various computational, instrument-based data and other types of resources. A Grid portal provides an end-user view of the collected resources available to the members of the VO.

Persistent Archive (PA)	Preservation environments handle technology evolution by providing appropriate abstraction layers to manage mappings between old and new protocols, software and hardware systems, while maintaining authentic records.
Mutual Authorization	Refines the CDC and NFC use cases by introducing the scenario of the job submitter authorizing the resource on which the job will eventually execute.
Resource Usage Service (RUS)	Facilitates the mediation of resource usage metrics produced by applications, middleware, operating systems, and physical (compute and network) resources in a distributed, heterogeneous environment.

Analysis of the use cases and other relevant input led us to identify both important and apparently broadly relevant characteristics of Grid environments and applications, and functionalities that appear to have broad relevance to a variety of application scenarios. We summarize our findings in the following sections.

2.1 Heterogeneous environment support

Some use cases involve highly constrained and/or homogeneous environments that may well motivate specialized profiles. However, it is clear that, in general, Grid environments tend to be heterogeneous and distributed, encompassing a variety of operating systems (e.g., Unix, Linux, Windows, or embedded systems), hosting environments (e.g., J2EE, .NET), and devices (e.g., computers, instruments, sensors, storage systems, databases, networks). Resource virtualization is essential to reduce the complexity of managing such heterogeneous systems and to handle diverse resources in a unified way. In addition, standard interfaces are required to facilitate interoperability. Moreover, many functions required in distributed environments (such as security and resource management) may already be implemented by stable and reliable existing legacy applications, and thus it is required to be able to integrate such legacy applications into the Grid.

The following functional requirements are related to this requirement.

- *Resource Discovery*.
  OGSA must enable discovering and identifying various kinds of resources in a consistent way, regardless of their implementation or vendor. In addition, it must be possible to retrieve their semantic information (e.g. interface, function, availability, location, policy, or other criteria) based on their identity.
- *Resource Query.*
  It must be possible to search for resources based on particular attribute information, regardless of their implementation or vendor.
- *Lifecycle Management of Hosting Environment.*
  Mechanisms are required for managing the lifecycle of heterogeneous hosting environments in a uniform and consistent way.

2.2 Resource sharing across organizations

One of the main purposes of Grid technology is to utilize resources transparently across administrative domains. Virtual Organizations (VO) provide a context that can be used to associate users, requests, resources, policies, and agreements without being limited by barriers

such as existing sites or organizations. Sharing resources across organizations implies various security requirements, some of which are listed here. Section 2.6 describes the security requirements in more detail.

The following functional requirements are related to this requirement.

- *VO Management.*
  OGSA must address creation and management of VOs, including manipulation of a VO's constituents such as users, requests, resources, policies, and agreements.
- *VO Security.*
  It is required to realize such functionalities as Identity Federation Services, which provide (federated) identity assertions for users or resources, and Authorization Authorities, which decide whether to permit requests based on attributes and policies that are applied to the requests.
- *Accounting (billing).*
  OGSA must provide mechanisms which allow for recording usage of shared resources, and allow owners of the resources, possibly belonging to a different organization from the resource users, to bill users for the usage of the resources.

Comment [JT1]: Needs some re-wording (e.g. OGSA must provide…)
Also, shouldn't this list be exhaustive as it's a requirements section? i.e. we shouldn't just give examples, we should state exactly what must be provided.

2.3 Resource utilization

Often resource allocation provides for the worst case scenarios (e.g., highest expected load, recovering from failures, etc) and leads to resource underutilization as resources allocated to address such scenarios remain unused for long periods. Mechanisms for monitoring resource usage, changing resource allocation, and adding or removing resources dynamically and on demand are required, so that system administrators can manage resources in an effective and well-planned manner.

The following functional requirements are related to this requirement.

- *Reservation.*
  It must be possible to guarantee that requested resources can be used for the specified period. A mechanism is required for configuring policies so that administrators can allocate resources in a well-planned way, for example, limiting the amount of resources available during a certain period, or the types of resources that may be used.
- *Metering.*
  OGSA must enable monitoring of resource utilization for purposes of cost allocation (i.e., charge-back), capacity and trend analysis, dynamic provisioning, service pricing, fraud and intrusion detection, and/or billing.
- *Monitoring.*
  OGSA must enable monitoring of both static and dynamic information about the state of resources, services, and applications. This information can be used for such tasks as fault detection, performance analysis, and job scheduling.
- *Logging.*
  OGSA should support basic log semantics, enable the exploitation of existing logger systems, provide a mechanism to persist log records, and support sequential writes at the granularity of an individual record, with each record representing one log artifact.

2.4 Job execution and QoS assurance

OGSA must enable submitted jobs to have coordinated access to VO resources, by automatically matching the requirements of the job with the available resources, while satisfying the resource allocation policies specified by the system administrator. Moreover, OGSA must provide functions such as monitoring the state of job execution, analysis and projection of resource usage, manageability of jobs and resources based, so that jobs can provide the desired QoS. The desired QoS is defined by an agreement between service requestor and provider.

It is expected that the resources allocated to a job are dynamically adjusted based on the workload. For example, in the Severe Storm Modeling use case, computing resources must be provisioned on-demand to satisfy the need to complete a forecast on time.

The following functional requirements are related to this requirement.

- *Scheduling.*
  OGSA must enable scheduling and executing tasks based on such information as specified priority and workload of resources. It is also required to realize meta-scheduling of resources across administrative domains.
- *Lifecycle Management of Job.*
  OGSA must provide support for job execution throughout the lifetime of the job, including functions such as starting/stopping the job, monitoring the state of job execution, and error detection. It is required to be able to manage jobs in an integrated way without regard to the physical location or the number of the resources, even in the case that a job is using multiple distributed resources.
- *Workflow Management.*
  Many applications can be wrapped in scripts or processes that require licenses and other resources from multiple sources. Applications coordinate using the file system or based on events. A Fusion Grid network service is a workflow of multiple components.
- *Service Composition.*
  Both orchestration and choreography create higher level and possibly cross-organizational services that combine existing services. Orchestration refers to an executable process that can interact with both internal and external services. Choreography combines services at message level and includes logic and task execution ordering.
- *Service Level Agreements.*
  Service level assurance based on agreements is required in various domains, not limited to Grid systems. OGSA must provide standard interfaces to create and manage agreements based on negotiation between service requester and provider.

Comment [JT2]: Need to explain this.

Comment [JT3]: What is the requirement for Workflow management and service composition? In both cases you explain what they are, but you don't document the requirement.

Comment [JT4]: Need to mention that this is a use cast.

Comment [JT5]: The list doesn't include a monitoring requirement, although this is mentioned in the intro text above.

2.5 Data Services and QoS assurance

Efficient processing of huge quantities of data is required in more and more fields of science and technology. In addition, data sharing is important, for example to enable sharing information stored in databases which are managed and administered independently. Also in business areas, archiving of data and data management are essential requirements. In the Online Media and Entertainment use case, it is a critical requirement to handle multimedia data (such as video, audio, and so forth) and to assure QoS of content delivery.

The following functional requirements are related to this requirement.

- *Data access.*
  OGSA must enable easy access to various types of data (such as streaming and cache) independent of its physical location, by abstracting underlying data sources. Mechanisms are also required for controlling access rights at data object level.
- *Data integration.*
  OGSA must provide mechanisms for integrating heterogeneous and distributed data. It is also required to be able to search data available in various formats in a uniform way.
- *Data provisioning.*
  The required data must be made available at the requested location. OGSA must allow for selection from various ways such as transfer, copying, and caching, according to the nature of data.
- *Data catalog.*
  OGSA must enable finding, invoking, and tracking data by means of metadata.

2.6 Security

Safe administration requires controlling access to services through robust security protocols and according to some security policy. Thus, authentication mechanisms are required so that the identity of individuals and services can be established, and service providers must implement authorization mechanisms to enforce policy over how each service can be used. Mechanisms are also required for integrating and interoperating with existing security infrastructures. In addition, standard, secure mechanisms are required which can be deployed to protect Grid systems while supporting safe resource sharing across administrative domains. Sharing of resources by service users requires some kind of isolation mechanism.

The following functional requirements are related to this requirement.

- *Authentication, Authorization, and Accounting.*
  Obtaining application programs and deploying them into a Grid system may require authentication/authorization. The Grid system may have to identify users' security policies. Authorization should accommodate various access control models and implementation.
- *Multiple Security Infrastructures.*
  Distributed operation implies a need to integrate and interoperate with multiple security infrastructures. OGSA needs to integrate and interoperate with existing security architectures and models which are typically difficult to be replace.
- *Firewall Traversal.*
  OGSA needs standard and secure mechanisms that can be deployed to protect institutions while also enabling cross-firewall interaction without compromising local control of firewall policy.
- *Isolation.*
  Various kinds of isolation must be ensured, such as isolation of users, performance isolation, and isolation between content offerings within the same Grid system.
- *Delegation.*
  Mechanisms which allow for delegation of access rights from requestors to services are required. The authority transferred through delegation is scoped only to the task(s) intended and within a limited lifetime to minimize the risk of misuse of delegated authority.

- *Policy Exchange*.
  OGSA must allow service requestors and providers to exchange dynamically security (among other) policy information to establish a negotiated security context between them.
- *Manageability*.
  Manageability of security functionality is needed such as identity management, policy management, key management, and so forth. The need for security management also includes higher-level requirements such as anti-virus protection, intrusion detection and protection.

2.7 Administrative cost reduction

The complexity of administrating large scale distributed, heterogeneous systems increases administration costs and the risk of human errors. Support for common administration tasks, by automating administrative operations and consistent management of integrated resources, is needed. Automation may be based on policies configured by administrators.

The following functional requirements are related to this requirement.

- *Provisioning.*
  Provisioning automates the complicated process of resource allocation, deployment, and configuration (see the next item) based on policies specified by administrator.
- *Deployment and Configuration.*
  It must be possible to deploy the required applications and data to resources and configure them automatically. If necessary deploy and re-configure hosting environments such as OS and middleware to prepare the environment needed for job execution.
- *Application Contents Management.*
  Mechanisms are required for managing complicated applications consisting of many components. These mechanisms must allow for change management of components in a concise and reliable manner, even without expert knowledge about the applications.
- *Problem Determination.*
  OGSA must provide mechanisms for problem determination in order for administrators to keep track of the state of the system and cope quickly with emerging problems. In particular, mechanisms for problem detection, cause analysis, impact analysis, and reference to past problem-handling are required.

2.8 Scalable computing

In addition to remotely accessing supercomputers at a geographically remote computer center, utilizing multiple supercomputers in order to solve complex problem is one key aim of e-Science grid system. Added value such as drastic reduction of job execution time, high availability, and scalability can be created.

OGSA must provide mechanisms for job execution on a great number of heterogeneous resources while ensuring the integrity of the entire job. Provisioning of a great number of resources distributed geographically, and starting/stopping and error-handling of jobs while maintaining the integrity of the entire job is required.

The following functional requirements are related to this requirement.

- *Distributed supercomputing.*
  Using distributed supercomputers simultaneously.
- *High-throughput computing.*
  Calculation using large numbers of computers including desktop PCs, e.g., SETI@HOME. OGSA should provide mechanisms for adjusting and optimizing parallel job execution in order to improve throughput of the entire computation process, as well as optimizing a single calculation process.

2.9 Availability

High availability is often realized by expensive fault-tolerant hardware and complex cluster systems. Because of the widespread use of IT systems to provide essential public infrastructure services, an increasing number of systems are required to have high availability while minimizing costs. It is required to realize systems of high availability at reasonable cost using components with longer or more unpredictable Mean Time To Repair (MTTR) than existing high-reliability systems.

In the CDC use case, if a Data Center becomes unavailable due to a disaster such as an earthquake or fire, a remote backup Data Center has to take over the affected application systems. And in the Service-Based Distributed Query Processing use case, fault tolerance is particularly important for long-running queries that can potentially return large amounts of data and would be too costly to restart.

Virtual Organizations enable transparent sharing of alternative resources *across* organizations as well as *within* organizations, and thus can be used as one building block to realize stable, highly-reliable execution environments at reasonable cost. In addition, policy-based autonomous control and dynamic provisioning are keys to realizing systems of high flexibility and recoverability.

The following functional requirements are related to this requirement.

- *Disaster recovery.*
  Operation of a Grid system must be recovered quickly and efficiently in case of natural or human-caused disaster, avoiding long-term service disruption. Remote backup and simplification or automation of system reconstruction at reasonable cost is required.
- *Fault Management.*
  Running jobs must not be lost because of resource faults. Mechanisms are required for fault detection, and diagnosis of causes or impacts on running jobs. In addition, if possible, automation of fault handling, such as migration of jobs onto alternative resources, is required.

3 Capabilities

The infrastructure elements just described provide powerful building blocks for stateful service-oriented systems. In particular, WSDL allows us to define service interfaces in a standard manner, facilitating service discovery and the creation of interoperable systems. WSRF mechanisms for representing and managing state have broad applicability, allowing us for example to define visible state for purposes of monitoring, create and manage transient stateful entities such as policies, and (with WS-Notification support) subscribe to and deliver notifications of changes to this state. WS-Security provides a basis for authentication.

However, these infrastructure elements certainly do not address all of the requirements identified in Section 2. Thus, we now turn to OGSA proper and describe the capabilities that are provided

by its various components. We divide its functions into four broad groups: core, data, program execution, and resource management services.

3.1 Execution Management Services

Needs to be updated according to May'04 F2F discussions.

Program execution services enable applications to have *coordinated* access to underlying VO resources, regardless of their physical location or access mechanisms. When an application utilizing the Grid makes use of more than one physical resource during its execution, Program Execution middleware maps the resource requirements of the user application to the multiple physical resources that are required to run that application. The Program Execution services are the key to making VO resources easily accessible to end-users, by automatically matching the requirements of a Grid application with the available resources, while staying within the conditions that the VO has specified with the underlying resource managers.

Interoperability is the key to building up a Program Execution Services infrastructure. In order to allow the higher order constructs such as Program Execution to work with the lower level resource managers, there must be agreement on how these entities will interact with each other, even though the lower level resource managers might be very different from each other, in function and in interface. In order to meet the requirement for interoperability, standards are required which define the interfaces through which resource managers are accessed and managed. Additionally, there is a requirement that the services representing the resource managers act using standard semantics, so that the behavior of the resource manager is predictable to the scheduler.

OGSA-based Grid environments may be composed of many different, interacting services. Each such service may be subject to different policies on how to manage the underlying resources. In order to deal with the complexities of large collections of these services, there need to be mechanisms for Grid service management and the allocation of resources for applications. One such mechanism is defined through the proposed *WS-Agreement* interface [4]. The specification document for WS-Agreement describes it as "… the ability to create services and adjust their policies and behaviors based on organizational goals and application requirements." WS-Agreement defines the Agreement-based Service Management model, which is specified as a set of WSRF-compliant portTypes allowing clients to negotiate with management services in order to manage services or other legacy applications.

3.1.1 Conceptual Frameworks

Workload realization is the general set of use cases that take workload requests and map them to appropriate resources within the Grid that can realize these workloads. Services manage and co-ordinate access to and consumption of geographically distributed resources by workloads realized on those resources.

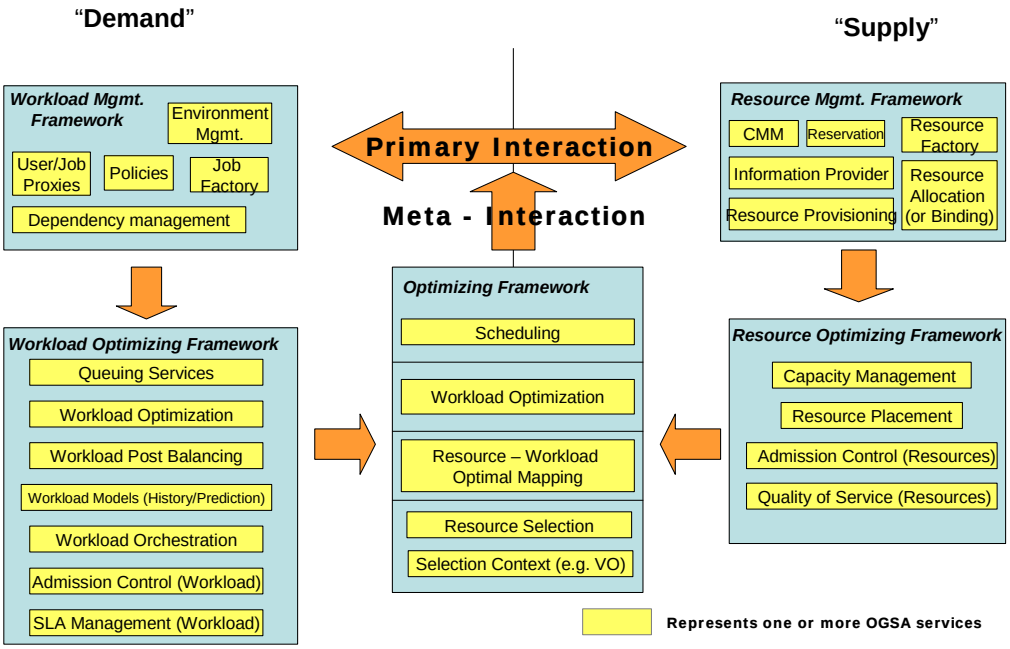

Figure 1: Grid Frameworks - Execution

The overall conceptual architecture can be visualized as shown in **Error! Reference source not found.**. Workload realization (or execution) can be visualized as a mapping between "demand" in the form of workloads and "supply" in the form of the available Grid resources. In a fundamental scenario, the system has to map the demand to the supply and provide the mechanism to realize these workloads on the resources. This primary mode can be augmented with other services, mechanisms and capabilities that provide alternate modes of interaction, including optimization of the mapping and scheduling a temporal and topological execution profile. In addition, other services manage and enforce the service level agreements with the user, and still other services tweak the resources and manage the available capacity to ensure that a desired quality of service is delivered. The alternate modes are not core to the execution framework, but can be added to augment the efficacy and efficiency of the overall system.

Services that belong to the *Resource Optimization Framework* are focused on the optimization of the supply side of the mapping. This can be done by admission control, resource utilization monitoring and metering, capacity projections, resource provisioning and load balancing across equivalent resources, and negotiation with workload optimization and/or management services to migrate workloads onto other resources so as to maximize resource utilization.

Services that belong to the *Workload Optimization Framework* are focused on the demand side of the mapping. These services may queue requests to prevent resource saturation and manage relative request priorities, or may perform post-balancing by migrating workloads to appropriate resources, depending on the potential to be penalized or rewarded for missing or exceeding SLAs respectively.

Services in the *Optimizing Framework* are focused on resolving any contentions that the myopic views of the respective resource or workload optimization frameworks may create. These services arbitrate and modulate the primary interactions either in an 'in-band' or an 'out-of-band' manner.

The term "resources" is used in its most general sense, and can include virtualized physical resources such as CPU, storage, memory, and/or virtual resources such as software licenses or data. As shown by **Error! Reference source not found.**, workloads are composite entities that have multiple levels of "execution entities": workloads are made up of jobs which in turn are made up of tasks which in turn are made up of tasklets. Each of these composite entities has a manager.

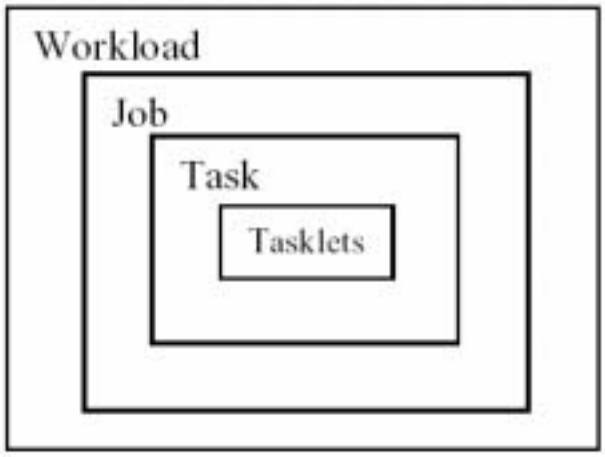

Figure 2: Workload, Job, Task, and Tasklet

A possible set of interactions among services to execute workloads are shown in **Error! Reference source not found.**. The services can be at multiple levels in the Grid and can be composed of other services or service groups.

Figure 3: Service interactions for workload execution (partial enumeration)

A physical organization of the services and one of the possible collaboration scenarios is shown in **Error! Reference source not found.**.

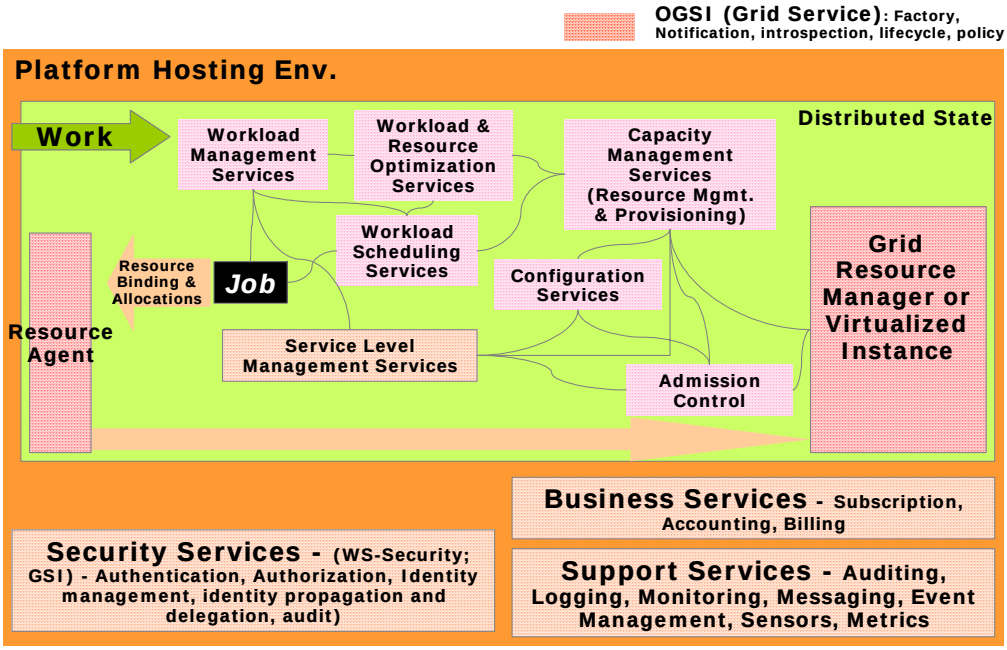

Figure 4: Example Grid execution service interaction

3.1.2 WS-Agreement Based Scheduler Instantiation

To put it in concrete terms, a user who wants to submit a compute job to run on a cluster would have their service client contact a job management service and negotiate a set of agreements that ensure that the job has access to required CPUs, memory, storage space, etc.

WS-Agreement defines fundamental mechanisms based on Agreement services, which represent an ongoing relationship between an agreement provider and an agreement initiator. The agreements define the behavior of a delivered service with respect to a service consumer. The Agreement will most likely be defined in sets of domain specific agreement terms (defined in other specifications), since the WS-Agreement specification is focused on defining the abstraction of the agreement and the protocol for reaching agreement, rather than on defining sets of agreement terms.

Referring back to the job submission example, the client might contact a job management service, which implements the AgreementFactory interface, with creation parameters that say “my job *must have* a software license for application X, *would like to have* 8 CPUs, and *would like to have* 4 Gbytes of RAM.” If the job management service couldn’t provide the software license (the “must have”), the agreement terms would be rejected. If it could provide all of the terms, an Agreement service instance representing the job resources would be created. If, because of available resource constraints, the job management service couldn’t fulfill the terms of the original creation parameters, but could supply either 4 CPUs and 4 Gbytes of RAM, or 8 CPUs and 2 Gbytes of RAM, the job management service could create an AgreementOffer which included two potential agreements: “app X, 4 CPUs, 4 Gbytes” and “app X, 8 CPUs, 2 Gbytes”, one of which the client could choose (because CPUs and memory were terms subject to counter-offers).

By defining various sets of terms for representing different types of resources available within the VO, Community Schedulers can be written which can negotiate with resource managers for the use of the underlying resources on behalf of the user community.

Program Execution services utilize WS-Agreement portTypes both for its client interface and for its interface to underlying resources, with the goal of allowing hierarchical VO deployment – e.g., one Community Scheduler talks to another Community Scheduler-based resource conglomeration. Another key aspect of agreement is to consider the constraints of the service provider based on Service Level Agreements in a business context, and their reflection in underlying resource manager policies. Not all constraints are based simply on the availability or unavailability of resources. Some are management policy specified, based on pricing, usage, higher priorities, and the like.

Error! Reference source not found. shows the services required for Program Execution, which services include:

- Agreement Factory Service – create agreement services based on domain specific terms such as job, reservation and data access terms.
- Job Agreement Service – creates, monitors and controls compute jobs.
- Reservation Agreement Service – guarantees that resources are available for running a job.
- Data Access Agreement Service – staging the required application and data.
- Queuing Service – provides a service through which administrators can customize and define scheduling policies at the VO level, and/or at the different resource manager levels.
- Index Service – allows for the propagation of information between resource managers and the metascheduler.

Program Execution services can be flexibly composed to offer a whole spectrum of time-to-result QoS types, ranging from online/interactive to batch with specific or no turn-around time requirements:

- *Online/interactive jobs* can be sent by the Job Agreement service to the Resource Managers (RMs) immediately, without getting queued.
- *Batch jobs* will be queued by the Queuing services and dispatched to RMs later. The jobs with specific turn-around time may use the available reservations made with the RMs at specific times so that their deadlines can be met. The jobs with no specific turn-around time can be queued at the VO level or at the backend RM level, until the required resource is available, or can backfill the existing reservations without delaying the start time of the more time-critical jobs.

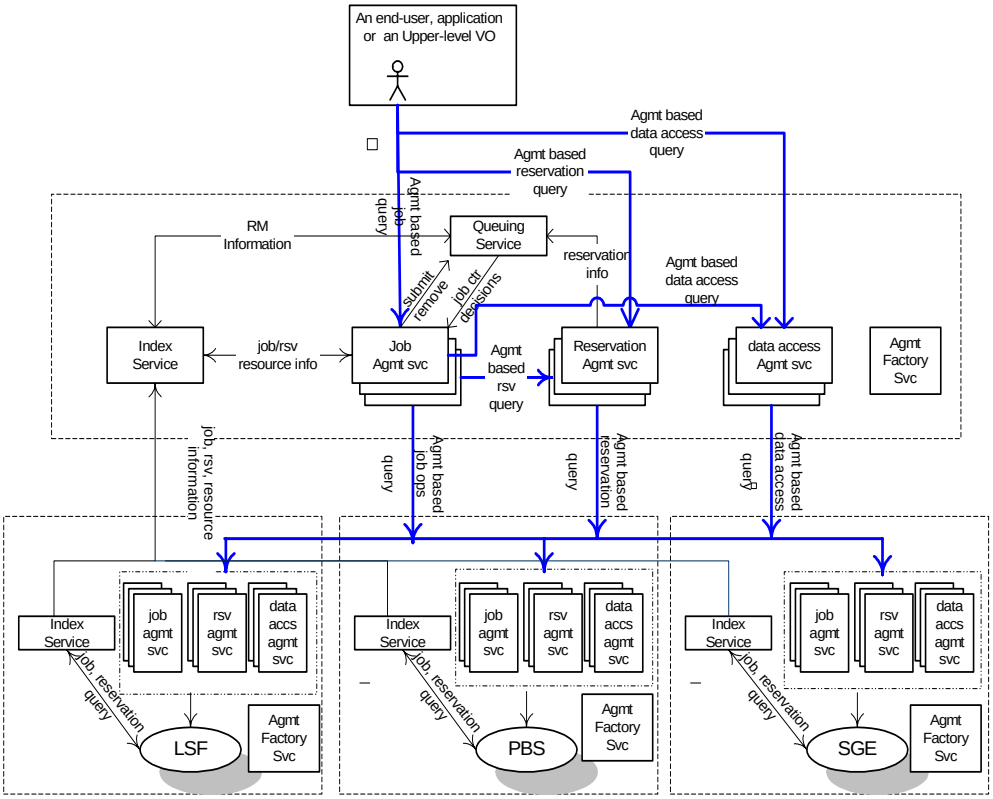

Figure 5: An Instantiation of Program Execution Architecture and Services

3.1.3 Instantiation with "Broker Service"

The Program Execution services must support the types of jobs. After a submitted job-request has been accepted, a ***broker service*** is used to select the resources (including the hosting environment) that are necessary to execute the job. [JT: First sentence in this para needs revision.]

After the pre-processing is completed successfully the requester is given a unique reference to the agreement. If necessary resources cannot be allocated, an error should be returned. Otherwise, the agreement can be used to query for information about the job as well as to control it – e.g., to modify job requirements or to cancel the job request.

The submitter can provide the following attributes or terms when submitting a job request.

- The service, program, or workflow that should be instantiated.
- The starting time of the job, including how long the job will execute.
- Resources, including data, needed by the job. (Resource descriptions could be given in XML.)
- User information.
- SLA information, including priority.
- Application startup script.

When the specified job start time arrives the scheduler must arrange for the job to start executing. If a startup script was specified then the script should be executed. A job may have its resources increased or decreased by the scheduler. Changes in resource allocation must be acknowledged synchronously. When a job terminates, the Program Execution services must carry out the following processing:

- Inform the client of the job's final status.
- Release the resources allocated to the job by the hosting environment.
- Delete the job from the scheduler's database.

The scheduling service is responsible for starting and managing all submitted jobs. Each job has a pre-determined start time, but priority and SLA attributes may cause it to be changed. The job state may be retrieved at any time.

Once the resource reservation period has finished then the job must be stopped and the resources used by the job must be returned to the resource management service. Jobs may be temporarily suspended or migrated based on the SLA (priority) associated with each job. Jobs with high priority are executed first.

Scheduling and resource reservation information associated with each job is kept in stable storage so that it is not lost in the event of failure. The scheduling service allows writing and reading of the scheduling data.

Resource reservation information is updated by the resource management service. In addition accounting information, such as which resources were used by the job, and for how long, is also recorded in the database.

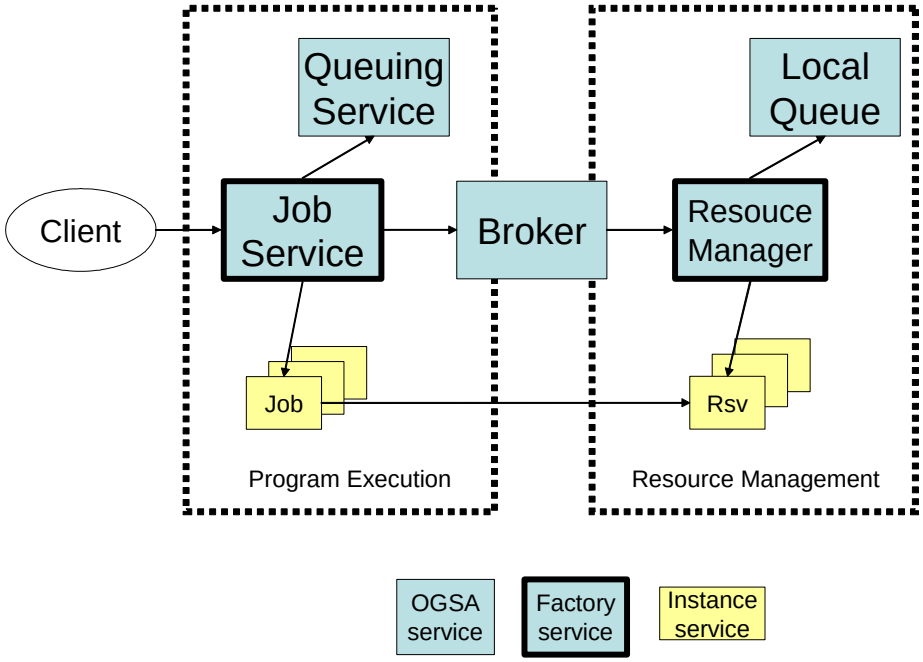

Figure 6: Program Execution with Broker

3.2 Data Services

Needs to be reviewed.

3.2.1 Objectives

The data management capabilities of OGSA provide the capabilities needed to move data to where it is needed, manage replicated copies, run queries and updates, and transform data into new formats. They also provide the capabilities necessary to manage the metadata that describes OGSA services, in particular the provenance of the data itself.

With the exception of certain classes of metadata, the data capabilities do not specify the meaning of any particular data. Some other OGSA services may use the data services and add meaning of their own; an example being the information services.

3.2.2 Models

3.2.2.1 Types of Data Resource

A data resource is any entity that can act as a source or sink of data. The heterogonous nature of the Grid means that many different types of data must be supported. These include (but are not limited to):

- *Flat Files*. The simplest form of data is a file with application-specific structure. For OGSA these files are opaque.
  Some file formats support database-like queries. Examples include comma-separated values, which can be queried like relational tables, and XML files, which can be queried using XMLQuery. The data access services support these and can also be extended to support specialised queries over new file formats.

- *Streams*. Potentially infinite sequences of data values are called streams. The data access services support queries and transformations over streams.
- *DBMS*. Several kinds of database management systems may be part of Grids. These include Relational, XML, and Object-oriented databases, among others.
- *Catalogues*. A catalogue structures and tracks other data services. A simple example of a catalogue is a directory, which lists a set of files. Nested directories are equivalent to a hierarchic namespace.
- *Derivations*. Some data is the result of asynchronous queries or transformations on other data. These derivations are managed like finite streams rather than single items.

Data Services themselves can be data resources for other services, as can sensor devices and programs that generate data.

3.2.2.2 Example Scenarios

Data management capabilities are fundamental to the Grid. This section describes a very few examples of how they can be used to support a range of activities.

- *Staging*. The simplest use of the OGSA data services is to move input data to a remote resource ready for an application to run on that resource, and then to move the result to an appropriate place.
- *Replication*. To improve reliability and to reduce access times, data can be replicated across the Grid.
- *Federation*. The OGSA data services can allow a service to query and update multiple data sources that are curated and maintained separately. Federation differs from replication in that the sites maintain different data.
- *Derivation.* The OGSA data services support the automatic generation of one data service from another.
- *Metadata.* Data that describes services or other data is fundamental to the Grid. In the simplest form, this is just another use of the OGSA data services. In addition, OGSA provides support for maintaining links between data and metadata.

3.2.2.3 Supply/demand meta-pattern

3.2.3 Functional Capabilities

3.2.3.1 Transparency, Virtualization and Layered Interfaces

A distributed system may contain data maintained in different syntaxes, stored on different physical media, managed by different software systems, and made available via different protocols and interfaces. This data may be stored locally or remotely; may be unique or replicated; be materialized or derived on demand. The OGSA data services can define virtualisations over these data: abstract views that hide these distinctions and allow the data to be manipulated without regard for them.

Conversely, some users may require the ability to exploit these distinctions. For example, they may wish to make use of a particular query language for a given database, or to specify the location of the data resource to use. They may have an application that requires native access to the data, or they may need to tune performance parameters of the data resource. To support such users, the OGSA data services provide manual control of operations such as data staging or specifying a particular data resource. They may also be extended to support resource-specific operations.

3.2.3.2 Client APIs

OGSA supports existing applications (and developers) who use standard APIs such as NFS, CIFS, JDBC, ODBC, ADO, POSIX IO and XQuery. API libraries map the existing calls to the messages sent to OGSA data services.

3.2.3.3 Data Management

The OGSA data services offer reliable data transfer from one location to another. This may be to create a copy of the original or to migrate the original completely. Data may be cached at a given location to avoid unnecessary additional transfers. Data caches can be configured in terms of (e.g.) lifetime and update consistency. Data may also be replicated between multiple copies, thus increasing availability through redundancy.

3.2.3.4 Queries

The OGSA data access services provide mechanisms for querying data resources. In simple cases these may run an SQL query over a relational database, an XML query over an XML database, or a regular expression over a text file. Other services may implement text mining over a set of documents, or distributed queries over federated databases.
Synchronous queries return the data in the response to a request, while asynchronous queries expose the derived data as a new resources. Services may also deliver the results of a query to a specified set of other services.

Distributed query processors analyse each query and create sub-queries to be run on the resources that comprise the federation. They may also determine where intermediate processing is done in order to minimise network traffic. As such they have some similarity to workflow enactment engines.

3.2.3.5 Transformation

Data services may themselves transform data. For example, they may convert data from one format to another, or filter it, before moving it. They may support stored procedures that execute within the service, making the service a form of container. These transformations may be instigated explicitly by certain operations, or they may be programmed to be triggered automatically in response to certain conditions.

3.2.3.6 Data Update

OGSA data services provide a range of mechanisms for updating data resources, depending on the semantics of the data resource. For catalogues, the operations include creation, renaming and deletion. For structured files and databases the operations include the update of entries. For streams and other files the operations are largely limited to appending new data.

When a data resource has replicated versions or is the source for derived data services, the updates may be propagated to the replicated or derived versions. In this case, and in the case where several clients are updating the same data resource, the services may implement various forms of consistency maintenance, e.g. to ensure that a client always sees the results of its own updates in any queries that it issues itself.

Update operations may be part of transactions. Transaction support is provided by other OGSA services.

3.2.3.7 Security Mapping Extensions

Database management systems often implement sophisticated security mechanisms. Some of these provide a large range of possible operations and access control at the level of individual elements. Therefore the OGSA data services support extensions to the standard OGSA security

infrastructure to allow users, operators and applications to access the greater control provided by such systems.

3.2.3.8 Data Resource Configuration

Data resources often provide sophisticated configuration options. These can be made available to clients via the OGSA data services. In addition, the services may provide additional operations for configuring the virtualisation of the resource provided by the service. For example, a relational data service might allow for specific tables from an underlying database resource to be made part of the data service's data virtualization, or for views on the underlying database to be made available as tables within that virtualization.

3.2.3.9 Metadata

Metadata services are data services that store metadata about services. OGSA provides support for maintaining links between OGSA services and the metadata that describes them. This support includes provision for maintaining the consistency of the metadata.

The metadata for data services may include information about the structure of the data, including references to the schemas that describe the data. For some services this is not practical, as the data resources include many schemas that are modified frequently, and in these cases schema information will be provided by the services themselves.

3.2.3.10 Provenance

Metadata for data services may also include information about the provenance of the data. This may be at the level of the whole resource or of its component parts, sometimes to the level of individual elements. This in turn requires the services or other processes that generate the data to also maintain the consistency of the metadata. Complete provenance information can allow the data to be reconstructed by following the workflow that originally created it.

3.2.4 Properties

3.2.4.1 Quality of Service

The OGSA data services can implement various levels of QoS, such as guaranteed delivery and referential integrity.

3.2.4.2 Coherency

When data is replicated, cached or derived, a range of coherency operations are available.

3.2.4.3 Performance

The OGSA data services are designed to minimize the copying and movement of data to the minimum. This is a key factor in the overall performance of the Grid.

The data transfer services use monitoring information such as bandwidth, utilisation patterns and packet size. This enables them to choose the best approach for moving a given data set to suit the agreed quality of service.

3.2.4.4 Legal and Ethical Restrictions

The OGSA data services have to operate within an environment where a variety of legal and ethical issues affect their operation. Personal data will be restricted in the persons that can access it and the operations that they can perform (confidentiality). Privacy concerns may limit the sort of data that can be revealed about individuals even when data can be released about groups.

Copyright restricts the copies that can be created of certain data. In the European Union, the similar "Database right" applies specifically to databases. The security mechanisms provided by OGSA allow these restrictions to be specified, but care must be taken when using the data services.

3.2.5 Interactions with the rest of OGSA

This section details the interactions between the data services and the other parts of OGSA.

3.2.5.1 Transactions

Data services are the classic example for transactions and many data resources provide independent transactional support. OGSA allows data services to be part of transactions. Support is provided for conventional ACID transactions and for two phase commit. In addition, "time warp" co-ordination can also be supported.

3.2.5.2 Logging

In addition to the usual OGSA logging capabilities, the data services use the logging services to support (non-local) logging.

3.2.5.3 Provisioning

In addition to the provisioning of storage space and of services themselves, data services also require provisioning support for uploading data sets to data resources. Some may also require support for uploading filters and cutters to a given data service.

3.2.5.4 Resource Reservation

Data services may need to reserve certain resources in order to operate. As examples, a file transfer will require storage space and network bandwidth, while a distributed query system may additionally require compute power in order to perform join operations.

3.2.5.5 Discovery

The data services may use the discovery services not just for registering services themselves, but also for registering the data sets that are stored by those services. They may also register the location of schema definitions.

3.2.5.6 Security

The security services support the mapping of OGSA identities and roles to resource-specific identities and roles. The virtual organisation management services similarly provide control of these mappings. The security services also support for checking the integrity of data transfers.

3.2.5.7 Network management

Network management can be crucial when planning the transfer of large amounts of data. It may be necessary to configure the network parameters to suit the amount of data to be transferred and the time constraints specified on those transfers.

3.2.5.8 Naming

The naming of data is a key feature of the OGSA data services. These use the OGSA three-level naming schemes. The physical name (WSRF end-point reference) includes the address where a data set can be found. The logical name identifies the data set regardless of its actual location (or locations, if it is replicated). The context-mapped name is a user-friendly short name that requires some context to disambiguate.

For example, in a replicated file system, a physical name may specify the location of a file on a particular machine. The logical name will identify the file in a location-independent buy

universal way, e.g. by including a UUID. The context-mapped name may be a user-friendly short file name that can be disambiguated by referring to the context in which it is used.

3.2.5.9 Workflow

The data access services are closely entwined with OGSA workflow. Workflows can instruct the data access services to send query results to third parties and to apply transformations as the data is moved. This requires intimate knowledge of the capabilities of the resource by the workflow services. Also, a call to a distributed query service may cause a variety of data movements and operations on other machines. The data service can provide information to the workflow manager to ensure that the workflow enactment takes full account of these effects on the network and other resources.

3.2.5.10 Notification

A particular use of the notification service is to externalise database triggers. Database management systems often provide a mechanism that specifies actions to be taken when certain conditions (triggers) are met. OGSA can easily cause these triggers to call the notification services.

3.2.5.11 WS-Agreement

As with other OGSA services, the data services use WS-Agreement to negotiate and record quality of service agreements and payment agreements.

3.3 Resource Management Services

Needs to be updated according to May'04 F2F discussions.

Service orchestration. These interfaces provide ways to describe and manage the choreography of a set of interacting services.

Administration. Standard interfaces for such tasks as software deployment, change management, and identity management.

Provisioning and resource management. Negotiation of service level agreements and dynamic resource allocation and re-distribution consistent with SLA policy, including mechanisms that allow clients and workflows to acquire access to resources and services at a particular (future) time.

Reservation and Scheduling Services. Reservation Services provide the mechanism to make resource reservations at a particular time and for a particular duration. Scheduling Services provide the mechanism to schedule tasks according to their priorities.

Deployment Services. Deploy necessary software (OS, middleware, applications) and data into the hosting environment.

3.4 Security Services

Needs to be updated.

OGSA security architectural components must support, integrate, and unify popular security models, mechanisms, protocols, platforms, and technologies in a way that enables a variety of systems to interoperate securely. A preliminary OGSA Security Architecture document, developed within GGF's OGSA-Sec working group seeks to address these goals in a manner consistent with the security model that is currently being defined for the Web services framework. An associated OGSA Security Roadmap document enumerates the security related specifications that will be needed to ensure interoperable implementations of the OGSA Security Architecture.

The security of a Grid environment must take into account the security of various aspects involved in a service invocation, as depicted in Figure 2 and discussed in the following.

As discussed in §**Error! Reference source not found.**, a Web service can be accessed over a variety of protocol bindings. Given that bindings deal with protocol and message formats, security functions such as confidentiality, integrity, and authentication fall within the scope of bindings and thus are outside the scope of OGSA proper.

Comment [ITF6]: This statement doesn't make sense to me.

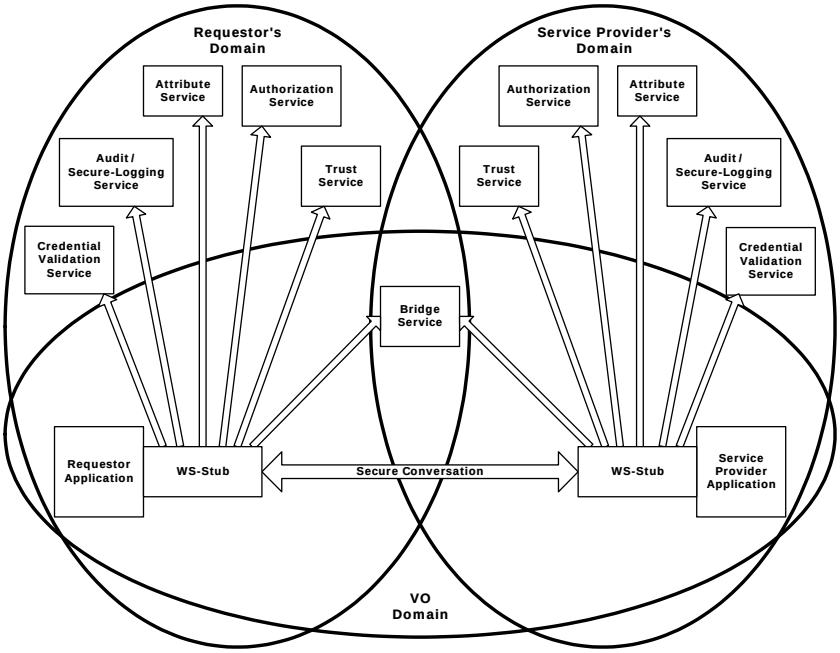

Figure 7: Security services in a virtual organization setting

Each participating end point can express the policy it wishes to see applied when engaging in a secure conversation with another end point. Policies can specify supported authentication mechanisms, required integrity and confidentiality protection, trust policies, privacy policies, and other security constraints. When invoking services dynamically, end points may need to discover the policies of a target service and establish trust relationships dynamically. (See §**Error! Reference source not found.** for more discussion of policy.)

Once a service requester and a service provider have determined each other's policies, they can establish a secure channel over which subsequent operations can be invoked. Such a channel should enforce the mutual agreed qualities of protection, including identification, confidentiality, and integrity. The security model must provide a mechanism by which authentication credentials from the service requester's domain can be translated into the service provider's domain and vice versa.

This translation is required in order for both ends to evaluate their mutual access policies based on the established credentials and the quality of the established channel.

OGSA's security model must address authentication, confidentiality, message integrity, policy expression and exchange, authorization, delegation, single logon, credential lifespan and renewal, privacy, secure logging, assurance, manageability, firewall traversal, and security at the WSRF

layer. We can expect that existing and evolving standards will be adopted or recognized in the Grid security model.

Error! Reference source not found. shows relationships between a requester, a service provider and many of the security services. Note that both requester and service provider are always subject to the security policies dictated by their respective administrative domains. Furthermore, a VO can have its own security policy that can enable the sharing of the submitted resources, but the associated rights will always be capped by the overruling resource-local policy. For many Grid applications, the resource owners and the individual requesters will not "know" each other as they live in different administrative domains while their interactions are dynamically discovered and brokered by scheduler services and such. This implies that trust has to be dynamically established through introductions, and the concept of the VO as a bridge is seen as an important tool to build these dynamic trust relationships.

All security interfaces used by a service requester and service provider need to be standardized within OGSA. Compliant implementations will be able to make use of existing services and defined policies through configuration. Compliant implementations of a particular security related interface would be able to provide the associated and possibly alternative security services.

3.5 Self-Management Services

Needs to be updated.

3.5.1 Service-level Attainment

3.6 Information Services

Needs to be updated.

3.7 Context Services

Needs to be updated.

3.8 Infrastructure Assumptions

Our goal in defining the Open Grid Services Architecture is to define a coherent and integrated set of components that collectively address the requirements identified above within the context of a service-oriented architecture. We must necessarily make assumptions about the underlying infrastructure on which we build if we are to make concrete statements about higher-level services. There are many examples of specifications that tried to be too abstract and did not achieve their goal. For example, CORBA 1.1 failed to achieve interoperability because service names depended on implementation. Thus, just as the designs of TCP, DNS, and other higher-level TCP protocols and services are informed by the properties of the IP substrate on which they build, so our design of OGSA is influenced by our choice of underlying mechanisms. In this section, we list and to some extent justify those choices.

The primary assumption is that work on OGSA both builds on, and is contributing to the development of, the growing collection of technical specifications that form the emerging Web services (WS) architecture [2]. Indeed, OGSA can be viewed as a particular profile for the application of core WS standards. We may this choice because of a strong belief in the merits of a service-oriented architecture and our belief that the Web services architecture is the most effective route to follow to achieve a broadly adopted, industry standard service-oriented rendering of the functionality required for Grid systems.

This choice of Web services as an infrastructure and framework means that we assume that OGSA systems and applications are structured according to service-oriented architecture principles, and that service interfaces are defined by the Web Services Description Languages (WSDL). For now, we assume WSDL v1.1, with a move to WSDL 2.0 planned once that latter specification is finalized. We also assume XML as the lingua franca for description and representation (although recognizing that other representations may be required in some contexts, for example when performance is critical) and SOAP as the primary transport binding for OGSA services. In addition, we seek to develop service definitions that are consistent with the interoperability profiles defined through the WS Interoperability (WS-I) process.

While thus working within a Web services framework, we do not take it as a given that Web services standards as currently defined meet all Grid requirements. In some cases, existing specifications may require modification or extension. Thus, OGSA architects have been involved in the definition of WSDL 2.0 and in the review of WS-Security and related specifications, and we identify in this document other areas in which extensions to existing specifications are desirable. In other cases, Grid requirements motivate the introduction of entirely new service definitions.

One key area in which Grid requirements motivate extensions to existing specifications is security. Security issues arise at various layers of the OGSA infrastructure. We use WS-Security standard protocols to permit OGSA service requests to carry appropriate tokens securely for purposes of authentication, authorization, and message protection. End-to-end message protection is required by some scenarios addressed by the OGSA infrastructure, and thus OGSA must also provide for higher-level protection mechanisms such as XML encryption and digital signatures in addition to, or in place of, point-to-point transport-level security, such as TLS and IPSec. In addition to message-level security, an interoperable and composable infrastructure needs security components themselves rendered as services. There are various efforts underway to specify service definitions for these security services. For example, an OGSA authorization service may use the proposed WS-Agreement standard along with evolving OASIS standards SAML and XACML to express security assertions and access control description. When and where appropriate, OGSA will adopt, or define those security services.

A key area in which Grid requirements have motivated new specifications—specifications that have relevance beyond Grid scenarios—is in the area of state representation and manipulation. In particular, we assume that we have available as building blocks the interfaces and behaviors defined by the WS Resource Framework (WSRF), the recently proposed refactoring of the Open Grid Services Infrastructure (OGSI) [9]. WSRF defines an approach to modeling, accessing, and managing state; to grouping services; and to expressing faults. These mechanisms (defined after a 18-month design process within the GGF OGSI-WG) all have a fundamental role to play in constructing Grid systems. In addition, WSRF is being strongly considered for use in a variety of resource modeling and systems management standards efforts, such as the OASIS WSDM Technical Committee. Thus, OGSA-related standards that build on WSRF are likely to be well positioned for composition with efforts from these standards bodies.

Finally, we assume notification/eventing capabilities such as those defined within the recently proposed WS-Notification specifications. These specifications, derived like WSRF from OGSI, define notification mechanisms that build on WSRF mechanisms to support subscription to, and subsequent notification of changes to, state components. (WS-Eventing specification provides similar mechanisms, but does not yet connect to WSRF.)

4 Security Considerations

This specification defines requirements for interfaces, behaviors, and models used to structure and achieve interactions among services and their clients. While it is assumed that such interactions must be secured, the details of security are out of the scope of this specification. Instead, security should be addressed in related specifications that define how abstract interactions are bound to specific communication protocols, how service behaviors are specialized via policy-management interfaces, and how security features are delivered in specific programming environments.

5 Editor Information

Ian Foster
Distributed Systems Laboratory
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
Phone: 630-252-4619
Email: foster@mcs.anl.gov

Hiro Kishimoto
Grid computing and Bioinformatics Laboratory
Fujitsu Laboratories
4-1-1, Kamikodanaka, Nakahara, Kawasaki City, Japan
Phone: +81-44-754-2628
Email: hiro.kishimoto@jp.fujitsu.com

6 Contributors

We gratefully acknowledge the contributions made to this specification by Takuya Araki, Jamie Bernardin, Dave Berry, Shel Finkelstein, Jeffrey Frey, Dennis Gannon, Andrew Grimshaw, Bill Horn, Kate Keahey, Tan Lu, Fred Maciel, David Martin, Jeffrey Nick, Benny Rochwerger, John Rofrano, Andreas Savva, Frank Siebenlist, Chris Smith, David Snelling, Ravi Subramaniam, Jem Treadwell, Jay Unger, Jeffrin Von Reich, James Warnes, and Ming Xu.

References

1. Handle System, 2004. www.handle.net.
2. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. Web Services Architecture. W3C, Working Draft http://www.w3.org/TR/2003/WD-ws-arch-20030808/, 2003.
3. Cabrera, F., Copeland, G., Cox, B., Freund, T., Klein, J., Storey, T. and Thatte, S. Web Services Transaction (WS-Transaction), 2002. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-transaction.asp.
4. Czajkowski, K., Dan, A., Rofrano, J., Tuecke, S. and Xu, M. Agreement-Based Grid Service Management (WS-Agreement). Global Grid Forum, Draft, 2003.
5. Foster, I., Gannon, D. and Kishimoto, H. Open Grid Services Architecture Use Cases. Global Grid Forum OGSA-WG, Draft draft-ggf-ogsa-usecase-1, 2003.
6. Foster, I., Kesselman, C., Nick, J. and Tuecke, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Globus Project, 2002. www.globus.org/research/papers/ogsa.pdf.

7. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications, 15* (3). 200-222. 2001.
8. Kahn, R. and Wilensky, R. A Framework for Distributed Digital Object Services. Corporation for National Research Initiatives, 1995. http://www.cnri.reston.va.us/home/cstr/arch/k-w.html.
9. Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S. and Kesselman, C. Grid Service Specification, 2002. www.globus.org/research/papers/gsspec.pdf.

Comment [JT7]: JT: This bibliography is maintained as an ENDNOTE add-in, which is not available to everyone. We need to decide on how to do this (any reason we can't just used a numbered list?). Item 11 (and any subsequent items) cannot currently be referenced.