Open Grid Services Architecture: A Roadmap

Abstract

Successful realization of the Open Grid Services Architecture (OGSA) vision of a broadly applicable and adopted framework for distributed system integration requires the early standardization of core services. The OGSA working group within the Global Grid Forum has been formed to develop a comprehensive and consistent OGSA roadmap that (a) defines, in broad but somewhat detailed terms, the scope of the services required to support both e-science and e-business applications, (b) identifies a core set of such services that are viewed as highest priority for definition, and (c) specifies at a high-level the functionalities required for these core services and the interrelationships among those core services. This draft document provides an initial outline for this roadmap.

1 Introduction

The Open Grid Services Architecture (OGSA) has been proposed as an enabling infrastructure for systems and applications that require the integration and management of services within distributed, heterogeneous, dynamic "virtual organizations" [1]. Whether confined to a single enterprise or extending to encompass external resource sharing and service provider relationships, service integration and management in these contexts can be technically challenging because of the need to achieve various end-to-end qualities of service when running on top of different native platforms. Building on Web services and Grid technologies, OGSA proposes to define a core Grid service semantics and, on top of this, an integrated set of service definitions that address critical application and system management concerns. The purposes of this definition process are twofold: first to simplify the creation of secure, robust systems and second to enable the creation of interoperable, portable, and reusable components and systems via the standardization of key interfaces and behaviors.

While the OGSA vision is broad, work to date has focused on the definition of a small set of core semantic elements. Specifically, the *Grid service specification* [3] being developed within the Open Grid Services Infrastructure (OGSI) working group of the Global Grid Forum defines, in terms of Web Services Description Language (WSDL) interfaces and associated conventions, the mechanisms that any OGSA-compliant service must use to describe and discover service attributes, create service instances, manage service lifetime, and subscribe to and deliver notifications.

While the Grid service specification defines essential building blocks for distributed systems, it certainly does not define all elements that arise when creating large-scale interoperable systems. We may also need address a wide variety of other issues, both fundamental and domain-specific, of which the following are just examples. 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? Without standardization in each of these (and other) areas, it is hard to build large-scale interoperable systems.

Given that the set of such issues is in principle large, it is important to identify those capabilities that are most critical so that specification effort can be focused in those areas, with the goal of defining, in a coordinated and timely fashion, a set of "core OGSA interfaces" that address the most urgent requirements.

2 Approach

We propose the following approach:

- Develop an initial draft for this roadmap that first provides a service laundry list and second proposes a small core set for early specification.
- Refine this draft roadmap via working group activities and public comment.
- Finalize an OGSA Roadmap v1 that identifies priorities for OGSA-related work.

In identifying services we can draw upon the following sources:

- GGF Grid Protocol Architecture document
- Globus Toolkit and related Grid services.
- UK eScience Architecture Roadmap (Malcolm Atkinson et al.)
- OGSA Security WG Roadmap.
- DAIS WG documents
- Data Grid architecture document.
- NPI documents.
- GridLab project's GAT.
- Unicore.
- TeraGrid.

3 OGSA Goals

OGSA exists so that we may build interoperable, usable Grids for industry, e-science and e-business. The Open Grid Service Infrastructure (OGSI) defines the extensions and refinements of the emerging web services standards that are needed to build grid services. These OGSI compliant web services, which we will call Grid Services, will be the components of future Grid infrastructure and application stacks. The job of OGSA is to build upon OGSI to define the specific set of "Core Grid Services" that are the essential components of every Grid.

The OGSA Working Group (OGSA-WG) has the following scope

- To define the actual core services and interoperability requirements that must exists between them.
- To produce and document the use cases that will drive our prioritization of core service features and mechanisms.
- To understand the protocols and bindings that are necessary but go beyond the scope of OGSI.
- To investigate the relationship between core service requirements and the hosting environments that will support them.

This is a large task. A simple definition of "Core Grid Services" is those things that must be part of every complete Grid implementation because if they are not there the application developers will have to write them themselves. However, there are many ways that such a set can be partitioned into real services. The OGSA-WG must develop a set of profiles of specific services and their interoperation and the composition mechanisms so that others may unambiguously implement them. The OGSA-WG will operate by spinning off other working groups, which will turn these profiles into precise specifications.

In addition to the broad goals described above, there are other, more specific goals for OGSA. These include

- Facilitating distributed resource management across heterogeneous platforms
- Providing seamless QoS delivery
- Building a common base for Autonomic management Solutions (OGSA provides an open, integrating infrastructure; Grid computing then addresses issues relating to accessing and sharing the infrastructure, while autonomic functions make it possible to manage the infrastructure and thus create self-configuring, self-optimizing systems.
- Providing a common infrastructure building blocks to avoid "stovepipe solution towers"
- Open and Published Interfaces
- Industry- standard integration technologies: web services, soap, xml, etc.
- Accomplished with a seamless integration with existing IT resources

4 OGSI Review

Remind people in a couple of pages what GSS is about. A list of interfaces and a description of their functionality.

5 Requirements Analysis

Our goal in this document is to identify those services that are fundamental to the realization of secure, reliable distributed systems, and/or of critical importance to major e-science or e-business applications. Ideally we would be guided in this requirements analysis process by a complete and well-defined set of use cases. In the absence of this information, we work from a less formal set of examples derived from applications with which we are familiar.

5.1 Target Environments

First make a few observations about target environments. Scientific. Business. Desktop. Others? (Alternatively, the use cases could be categorized in this way.)

It is important to bear in mind that the constituency for OGSA specifications is large and diverse, encompassing both a range of industrial participants and numerous "e-scientists" from the research and academic communities. This diversity is a substantial strength of the OGSA process, but also means that care must be taken when developing specifications to ensure that significant interests are not neglected.

5.2 Use Cases

The initial meetings of OGSA explored a number of detailed use cases for Grid applications. We organize these into two general categories: Scientific Applications and Commercial Grid Scenarios. However, we note that there is significant cross-over between these.

5.2.1 The Scientific Application as Grid Service (based on Kate Keahey's Fusion Collaboratory example and Gannon's discussion of weather prediction.)

In large scientific collaborations, it is common to have certain applications that are tied to a specific set of high performance computing resources. These applications are hard to port and maintain and they are updated frequently to capture improved science and algorithms. However they must be able to be run on-demand from authorized members of the user community and the users must be able to trust their performance and behavior. In some cases they are run for very long periods (many hours) to obtain accurate results and in other cases they may be run for only short periods to obtain partial results.

As a Grid service the factory pattern is the best model to design such an application service. Users would contact a factory service and provide details about parameters and input files and execution requirements. After authenticating the user, the factory can then contact resource broker services, data staging services and scheduling services to provide the user with a service contract specifying a set of possible time windows and performance guarantees that can be met to meet the users requirements. Once such a contract has been established, the factory creates an instance of a transient service that executes the application on behalf of the user. For long running applications, this transient service may mediate the interaction of a group of user clients with the running application for the purpose of monitoring or steering.

Instrumentation Grids. In some applications, such as predicting severe weather, a Grid of sensors spanning a wide area generate streams of data which are tied, in real-time, to large simulations and data mining tools running on remote supercomputers. As the storm evolves, the simulations trigger other applications to be run to predict more localized weather behavior such as tornados. The requirements of this application involve complex, on-demand resource provisioning and scheduling because the analysis must run in better-than-real-time speeds. It also involves the wide-area collaboration of many people using visualization clients that may interact directly with the workflow as it progresses. Autonomic processes must insure that the instruments stay operational and some instruments may need be retargeted automatically as the system evolves. Similar autonomic processes must monitor the progress of simulation and data analysis tasks to provision additional resources in case the existing allocation is not sufficient.

5.2.2 Commercial Grid Scenarios (based on examples from Hiro Kishimoto of Fujitsu, Jeff Nick of IBM and Andrew Grimshaw of Avaki)

Some commercial Grid applications have similarities to the scientific applications described above. For example, for Fujitsu, the application may be a simple Java program run as a EJB, but it may also require advanced resource reservation and dynamic rescheduling. This will require Service Level Agreements (SLAs) between the customer and application service provide to assure the client is satisfied with the quality of service.

Workflow Management. Another common batch processing activity encountered by both Fujitsu and Avaki involves legacy workflow management, which coordinates the execution of multiple jobs. Grid Service based workflows based on emerging web service workflow standards will allow multiple services to be composed into a single service. In this case, the workflow is itself a Grid service and the workflow engine may be distributed across multiple resources.

E-utilities. An e-utility is an on-demand Grid capability that is analogous to the water company or the electric power grid or telephone utilities. It is trusted and highly available and has autonomic functionality that keeps it almost always running. It is a service that allows you to pay for what you consume. There are two ways to think about and design e-utilities. One approach is to view it as a vertical e-utility. In this case, the e-utility is specific to a particular grid application, such as on-line multi-user gamming or a service specifically designed for a particular industrial application. The second case is that of the horizontal business service e-utility. Examples here include services and applications that cut across market sector such as business directories, business-to-business brokering services. Another example might be a media e-utility or a portal e-utility. Horizontal e-utilities might involve the infrastructure for virtualization and management of distributed computing resources that can be leveraged internally by IT organizations.

Data Federation. Another important use-case encountered by Avaki involves federating data archives that are stored at multiple sides belonging to an enterprise. In these cases the data consists of a combination of flat files and relational databases. Much of the data changes over time, i.e. there are frequent updates. Users and applications need access to all authorized data. Performance is critical, but the data must "stay at home". Coherence is critical, so caching must be done with great care. Audit trails must exist for all data updates.

Enterprise Collaborations. Avaki has seen several cases where multiple enterprises need to collaborate. For example, one enterprise has a genomics group in Raleigh, a server farm in Cambridge and a proteomics group in San Diego. At any given time they may have several partnerships with several other enterprises that may involve data subscriptions or licensed data or basic research contracts. In all cases the nature of the collaboration, and in may cases the very existence of the collaboration, must be kept secret. In some cases applications are shared as source code and in other cases applications are accessed only by remote, authorized clients. In other cases multiple applications, one from each enterprise, must be coupled together over the Grid and work as a single distributed workflow application. This type of enterprise application integration is seen in many different industries. This is like supply chain management – but with component simulations and data sets. The coupled application components are often proprietary and run in different companies and use different data sets stored in different companies.

5.3 Use-Case Issues

The OGSA-WG identified a dozen issues that came up often in this initial set of use-cases. They are

1. **Workflow management**. Almost all of the most demanding use-cases involve the ability to express the interaction of a number of services and to cast the composite activity into a single transient service instance working on behalf of a client or set of clients.
2. **Scheduling of service tasks**. Long recognized as an important capability for any information processing system, scheduling becomes extremely important and difficult for distributed Grid systems.
3. **Disaster Recovery**. As we begin to build complex distributed Grid infrastructure, disaster recovery becomes a critical capability. For distributed systems, failure must be considered one of the natural behaviors and disaster recovery mechanisms must be considered an essential component of the design. Autonomous system principles must be fully embraced as we design Grid applications and they should be reflected in the OGSA.
4. **Provisioning**. Computer CPUs, applications, licenses, storage, networks and instruments are all Grid resources that require provisioning. Others new types of limited resources will be invented and added to this list. OGSA will need a framework that will allow resource provisioning to be done in a uniform, consistent manner.

5. **Data Sharing**. Data management and sharing is one of the most common and important uses of Grids. How do we manage data archives so that they may be accessed across a Grid? How do we cache data and manage its consistency? How do we index and discover data and metadata? These are all questions that are central to most current Grid deployments. They are likely to become more important in the future.
6. **Legacy Application Management**. Legacy applications are those that cannot be changed, but they are too valuable to give up or to complex to rewrite. Grid infrastructure has to be built around them so that they can continue to be used.
7. **Vertical Utility Grids**. Some Grids are built as vertical utilities to service specialized user communities. For example, butterfly.net provides an enterprise Grid for multi-user distributed game playing.
8. **Horizontal Utilities**.
9. **Services Facilitating Brokering**. Many of the use cases require brokering services that can automate the process of selecting the appropriate resources for an application. OGSA will need a model for negotiating with brokers and a standard service model for building them.
10. **Application and Network-level Firewalls**. Many use cases require applications to be deployed on the other side of firewalls from the intended user clients. Inter-Grid collaboration often requires hoping institutional firewalls. OGSA will need standard, secure mechanisms that can be deployed which protect institutions but enable cross-firewall interaction.
11. **Virtual Organizations**. One of the main purposes for building Grids is to facilitate the interaction of a group of collaborators as a Virtual Organization (VO) that need share resources in a secure manner. OGSA will need mechanisms to create VOs and to enable the constructions of Grid Services that support the VO.
12. **CPU scavenging** is an important tool for an enterprise or VO to use to aggregate computing power that would otherwise go to waste. How can OGSA provide service infrastructure that will allow the creation of applications that use scavenged cycles? For example, consider a collection of desktop computers running software that supports integration into processing and/or storage pools managed via systems such as Condor, Entropia, United Devices, etc. Issues here include maximizing security in the absence of strong trust.

These issues are tied to a number of other very basic problems that must be solved by any Grid system. 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 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?

5.4 Initial OGSA Service Candidates.

The OGSA-WG considered a set of possible Core Grid Services that arise from the use-case issues described in the previous section. We list those here with reference to the use-case issues above.

Discovery Services are a cornerstone of any distributed system. This provides the ability for a client or another service to discover other services that make up the Grid. This is important for issues 2, 3, 4, 5, 8 and 9.

Registry Services provide the mechanisms for services to advertise their existence. Closely related to Discovery, this is also important for 2, 3, 4, 5, 8 and 9, and also 7, 11 and 12. It is already specified as part of OGSI.

Directory (file system) / Name Space Management Services provide a uniform way to identify and access both services and data objects and resource on the Grid. This is important for issues 1, 2, 3, 4, 5, 8, 9 and 11.

Authentication Services allow the Grid to recognize users. This is an essential component of all Grids and significant for 2, 4, 5, 7, 8, 9, 11 and 12.

Authorization Services allow Grid services to know when an authenticated user has the rights to access that service or resource. This is important for issues 2, 4, 5, 7, 8, 9, 11, 12.

Security protocol mapping services *e*nable distributed security protocols to be transparently mapped onto native platform security services for participation by platform resource managers not implemented to support the distributed security authentication and access control mechanism.

Resource Services provide an interface to a resource, such as a cpu, disk, network, management tools, etc. The Common Resource Model (CRM) provides a standard way to describe them. This is important for 2, 4, 5, 7, 8 and 12.

Reservation Services provide the mechanism for clients and workflows to acquire access to resources and services at a particular time. They are essential for 1, 2, 4, 9 and 12.

Brokering Services would mediate the SLAs that allow clients predicable access to Grid resources. This is significant for 1, 2, 4, 5 and especially 9.

Scheduling Services are closely related to Reservation and Brokering and constitute issue 2.

Load Balancing Services are services that maintain the integrity of SLA and performance contracts. They are clearly important for issues 2, 4, 7, 8, 9 and12,

Fault Tolerance Services would provide mechanisms to assure Grid functionality is maintained when rapid changes may take place in the Grid topology or load. This is closely related to issue 3 as well as 2 and 5. This is also related to a possible **Checkpoint and Restart Service**.

Event and Notification Services provide the tools to allow application and core services to communicate significant changes of state. For example, when an application services is created and registers itself with a Registry Service, the Registry may notify the Discovery Service that the new application service is available. This is important for 1, 2, 3, 7, 8 and 9. This service is also significant for Fault Tolerance services mentioned above and Logging and Accounting described below.

Logging Services maintain a record of the Grid state changes. They play a central role in autonomic behavior and disaster recovery as well providing an audit trail for various parts of Grid system failure or even performance tuning. This is important for 3, 7, 8, 9, 10, and 12.

Instrumentation and Monitoring, which keeps track of how well services and resources are performing. **Monitoring services**, supporting the discovery of “sensors” in a distributed environment, the collection and analysis of information from these sensors, the generation of alerts when unusual conditions are detected, and so forth. This is also a key component of Fault Tolerance. These are all part of a general class of *Problem determination services for distributed*

computing, including dump, trace, and log mechanisms with event tagging and correlation capabilities**.**

Accounting Services provide the mechanisms for service provides to be paid for authorized use of their resources. Accounting/auditing services, supporting the recording of usage data, secure storage of that data, analysis of that data for purposes of billing, fraud and intrusion detection, and so forth.

Data Caches and Data Replication Services provide a critical component for managing many of the issues related to Data Grid problems. This is at the heart of issues 5 and 11.

Metadata Search Services are a type of Discovery service to aid clients of Data Grids. They help locate/index the information about things. These are closely relate to the **File/DBMS services** and possible **Federated Data Management** services that are used as part of a vertical utility Grid. These are both critically important to issue 5.

Schema Reconciliation Services allow different data, service and policy schema to be reconciled so that the services can interact correctly. This is important for use-case issues 1, 2, 4, 5 and 9.

Transaction Services are an essential part of any Grid application that must update persistent information such as data archives or databases. It provides the appropriate multi-phase commit protocols to assure all parties that transactions have executed correctly in a distributed environment. This is important for use-case issues 1, 2, 3, 5, 7, 8, 9, 11 and 12.

Application Factory Services may not be core Grid Services but their basic interfaces may need to be standardized by OGSA. This relates to application deployment and provisioning as well as issue 6.

Clustering and Collection Services allow services to be grouped into related units. It also refers to services that manage group membership in general. Clustering services enable grouping and management of distributed peer service instances in order to provide coordinated management actions such as disaster recovery and load balancing, through dynamic join/leave semantics and ordered message and event delivery. A related concept is that of **Service Domain** which is discussed in greater detail below. Clustering and Collection is a component of issues 1, 2, 3, 5, 8, 9 and 11.

Policy Management Services allow Grid service providers the ability to specify and deploy service use policies. Such a policy may provide provisions for defining how a collection of services interacts or how resources are provisioned and SLAs are granted. This is significant for issues 2, 5, 8, 9, 11 and 12.

Workflow Engine Services provide ways to describe the choreography of a set of interacting services that are part of a Service Domain. *Workflow services*, supporting the coordinated execution of multiple application tasks on multiple distributed Grid resources. The workflow engine appears as a service instance to a client. It mediates the interaction between sub-services and handles the exceptions and faults that may occur in the workflow execution.

Context Services provide a container for the information related to user/service session interactions. This information may include user capabilities, service profiles, and applicable policy specifications. This is closely related to the emerging WS-Coordination specification that "enables an application service to create a context needed to propagate an activity to other services and to register for coordination protocols. The framework enables existing transaction processing, workflow, and other systems for coordination to hide their proprietary protocols and to operate in a heterogeneous environment."

Administration Services allow Grids to have a standard administration interface.

6 Basic OGSA Structure

Given this background on use case issues and the possible candidate services, the next concern is the architectural organization that is needed to build these services. In this section we consider some of the basic structure that must lie at the foundation of the OGSA.

6.1 Core Transport and Security

We note first that no Grid service execution or communication can occur without basic transport and security functions. These functions are defined within the Grid service specification as binding properties, meaning that a particular service implementation may choose to implement them using any protocol. Nevertheless, there must be some agreement on behaviors within any particular community, otherwise interoperability cannot be achieved.

6.2 Hosting Environments

Standard interface definitions such as those defined within the Grid service specification allow two services to interoperate. They do not address the portability of service implementations. Work is required to define standard hosting environments in order to enable portability. The following are just examples:

- Within a J2EE environment, standardized Java APIs can be defined to allow for portability among OGSI-enabled J2EE systems.
- Entropia, United Devices, and Condor allow untrusted (and untrusting) desktop systems to participate in distributed computations. A standard "desktop" hosting environment would allow for interoperability among these different systems.
- The TeraGrid project is defining standard "execution environments" for computers that run scientific applications. These execution environments assume Linux and define conventions for the locations of key executables and libraries, and for the names of certain environment variables.

6.3 Basic Interfaces and Behaviors

We list first a set of interface and behavior definitions that appear particularly fundamental to the creation of interoperable Grid systems. The inclusion of an interface in this list is not in any way a binding categorization: rather, it represents a value judgment concerning priorities.

- *Common resource models conformant to the OGSI service model.* The expression of underlying instrumented resources as OGSA services enables consistent distributed management and access to these resources without having to understand the details of implementation of the resources, whether they are instrumented in CIM or SNMP or MDS/Glue, etc.
- *Registry and service discovery*. OGSI defines a Registry interface and associated service data elements to support the registration, and subsequent discovery, of service instances. One or more standard registry behaviors need to be defined to permit service discovery in various settings.

- *Handle mapping*. OGSI defines a HandleMap interface to support the resolution of Grid service handles (GSHs) to Grid service references (GSRs). One or more standard HandleMap behaviors need to be defined to permit GSH resolution in various settings.
- *Service domain*. In what seems likely to be a common architectural approach, an OGSA-compliant "service" is implemented via a collection of internal services that are managed in some coordinated fashion. Standard interfaces and behaviors need to be defined to facilitate the creation and operation of, and the integration of new services into, such *service domains*.
- *Policy.* A Policy is a definitive goal, course or method of action based on a set of conditions, to guide and determine present and future decisions. Policies are implemented or executed within a particular context (such as policies defined for security, workload management, network quality of service, etc.). They provide a set of rules to administer, manage and control access to Grid resources. Policy Services are required to provide a framework for creating, managing, validating, distributing, transforming, resolving, and enforcing policies within a distributed environment.
- *Security*. Requirements here are wide reaching, encompassing policy services. Fortunately, a substantial effort has already started within the OGSA Security WG on an OGSA security roadmap that defines requirements, relationships to other standards efforts (e.g., WS Security) and priorities for early development.
- *Provisioning and resource management*. Negotiation of service level agreements and dynamic resource allocation and re-distribution consistent with SLA policy.
- *Distributed data management services*, supporting access to and manipulation of distributed data, whether in databases or files [2]. Services of interest include database access, data translation, replica management, replica location, and transactions.

7 Detailed Analysis

7.1 Common Resource Models / Resource Instrumentation

A *common resource model* is an abstract representation of real IT Resources, such as node, interface adaptor, disk, filesystem, IP address. It is also an abstract representation of logical IT Resources, that is, compositions of real IT Resources to build services and complete business applications.

Resources, either real or logical, define information that is useful for managing a resource – a concept known as *manageability*. Manageability details the aspects of a resource that support management including the *instrumentation* that allows an application or management tool to interact with a resource. *Management* is the active process of monitoring, modifying, and making decisions about a resource including the capabilities that use manageability information to perform activities or tasks associated with managing IT resources.

Manageable resources are exposed as Grid services in OGSA. A manageable (resource) Grid service implements the GridServices portType plus additional portTypes for the purpose of being used from or included in an application or management tool. Query of a resource's manageability information is through use of the GridServices portType's find and query operations. The use of additional portTypes provides manageability interfaces (common portType operations) to facilitate traditional systems management disciplines such as performance monitoring, problem determination, configuration, operational support, event notification, discovery, and lifecycle management.

Resources possess a lifecycle – an ordered set of states and the transitions between the states that a resource (in CRM, a service) goes through. Resources exist from the time they are installed until they are destroyed and a variety of states in between. Resources can be (and in most cases, are) managed differently over their lifetime. The resource lifecycle extensions describe the meaningful lifecycle states and transitions for the service, i.e., port types, operations, and service data. An application or management tool uses a resource's lifecycle state to better manage that service.

The resource models are expressed in XSD and embodied in a grid service. So, accessing the manageability information of a resource is just like as accessing any other grid service. The resource's manageability information can be instrumented using any instrumentation type of choice, such as CIM, SNMP, and LDAP. The resource model and grid service for that resource is independent of the underlying service implementation and resource instrumentation. The Common Resource Model (CRM) is not a strict algorithmic mapping for any one model. Existing models are mappable to CRM; those existing models with their operations and resource instrumentation can be service implementations of CRM.

7.2 Service Domains

The value of Grid solutions will be realized through the formation of Grid service collections and automated interactions between services and across collections. The OGSA Service Domain architecture proposes a high level abstraction model to describe the common behaviors, attributes, operations and interfaces to allow a collection of services to function as an integral unit and collaborate with others in a fully distributed, heterogeneous, but grid-enabled environment. The model includes, but is not limited to, the registration, discovery, selection, filtering, routing, fail-over, creation, destroying, enumeration, iteration, and topological mapping of service instances represented by the collection, as well as intra and inter collection interactions.

Service Domains represent multiple system and service objectives: resource oriented, such as CPU, storage, and network bandwidth; infrastructure oriented, such as security, routing, management; or application oriented, such as purchase orders, stock transactions, travel; etc. Domains can be homogeneous or heterogeneous, compute-intensive such as financial calculations or scientific and engineering computing, and transactional and business- process functions such as ERP, CRM; etc. Multiple Service Domains could be composed, nested, peer-to-peer, layered, overlapped, or mixed to satisfy the grid requirements for an enterprise complex, which could be part of a larger grid complex of heterogeneous business entities.

The architecture starts with the proposed OGSI ServiceCollection portType which extends the base GridService portType. ServiceColleciton defines the abilities to register (add) and unregister (remove) service instances from the service collection. The Service Domain model extends the ServiceCollection portType to provide a rich set of behaviors (and associated operations and attributes) on top of the collection of services with layered abstractions. A proposed set of behaviors that a Service Domain abstraction model should address initially include:

Filter: Behavior that supports choosing/allowing a Grid service to be included as part of a service collection.

Selection: Behavior that supports choosing a particular instance or a subset of instances within the service collection.

Topology: Behavior that supports a topological sort of the services in a service collection to impose one or more orders on the services within a service collection

Enumeration: Behavior that enumerates the services in a service domain.

Discovery: Behavior that allows a service domain to discover services from one or more registries and/or service domains to include as part of the service collection.

Policy: Behavior that allows policies to control the behavior of service domain operations as well as the constituent services (service domains).

In a large grid complex, service domains may be interconnected into a topology to meet the objectives of the complex. Each domain is itself a grid service representing a collection of services with specific functional objectives. Its intelligence comes from the rules and information fed by others together with the data gathered itself. Domains automatically perform the function for the grid complex by responding to the messages they receive.

7.3 Policy

A Policy is a definitive goal, course or method of action based on a set of conditions, to guide and determine present and future decisions. Policies are implemented or executed within a particular context (such as policies defined for security, workload management, network quality of service, etc.). They provide a set of rules to administer, manage and control access to Grid resources. The Policy Service provides a framework for creating, managing, validating, distributing, transforming, resolving, and enforcing policies within a distributed environment.

The proposed architecture for policy services is derived from the IETF draft policy framework architecture. It uses the IETF/DMTF CIM Policy Core Information Model, (IETF RFC3060 and DTMF DSP0108) as its information model. Policies are encoded and stored in an XML schema, which is derived from the CIM Policy Core Information Model extensions (PCIMe). The policy model provides policy rules that consist of conditions and actions. It also provides grouping for categorization / classification, and a scoping of policies by management discipline and role. Through its GridService behavior, it provides subscription to, and notification of, policy state changes so clients can be notified when policies become effective or expire, or are added, updated or deleted.

The Policy Service components consist of a **Policy Manager** which controls access to the **Policy Repository** for the purpose of policy creation and maintenance, **Policy Enforcement Points** that carry out the enforcement of policies on a specific device, and *discipline neutral* **Policy Agents** that can assist Policy Enforcement Points with policy retrieval, transformation and conflict resolution. To assist with these tasks, it defines a framework for **Policy Transformation** services to translate polices from business level objectives down to device level configurations, **Policy Validation** services which can be called by administrative tools and autonomic managers to validate policy changes, and **Policy Resolution** services to build higher level policy resolvers which could evaluate “value to the business” concerns as those expressed in a Service Level Agreement when resolving policy conflicts. All of these services are exposed as GridServices conforming to the Grid service specification.

7.4 Security

OGSA security architecture 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. The security architecture defined in the OGSA Security Architecture document (submitted to the OGSA SEC WG) is intended to be consistent with the security model that is currently being defined for the Web services framework used to realize OGSA’s service-oriented architecture.

The security of a Grid environment must take into account the security of various aspects involved in a Grid service invocation. This is depicted in the following Figure.

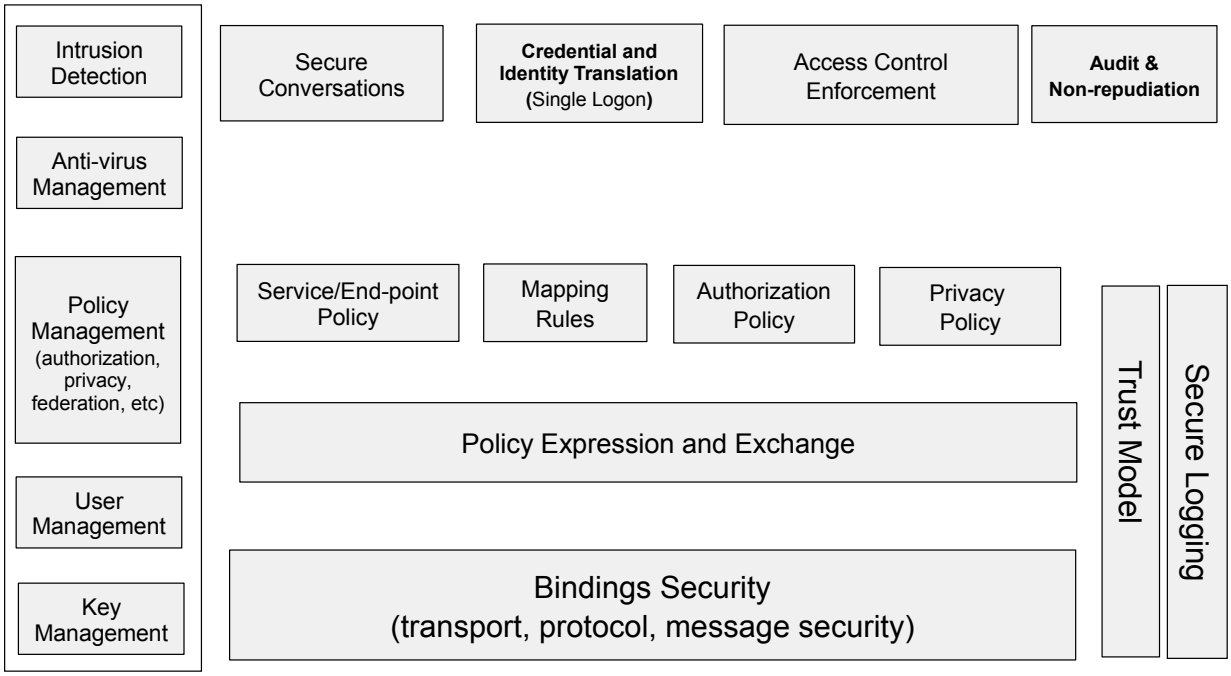


Figure 2: Components of Grid Security Model

A grid service can be accessed over a variety of protocols and message formats it supports. Given that bindings deal with protocol and message formats, they should provide support for quality of service, including such security functions as confidentiality, integrity, and authentication.

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, trust policies, privacy policies, and other security constraints. Given the dynamic nature of Grid service invocations, end points will often discover the policies of a target service and establish trust relationships with it dynamically.

Once a service requestor 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 various qualities of service including identification, confidentiality, and integrity. The security model must provide a mechanism by which authentication credentials from the service requestor'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.

Therefore, OGSA security model must address the following security disciplines: 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 OGSi layer.

As illustrated in the building blocks of Grid security model, and described above, existing and evolving standards will be adopted or recognized in the Grid security model.

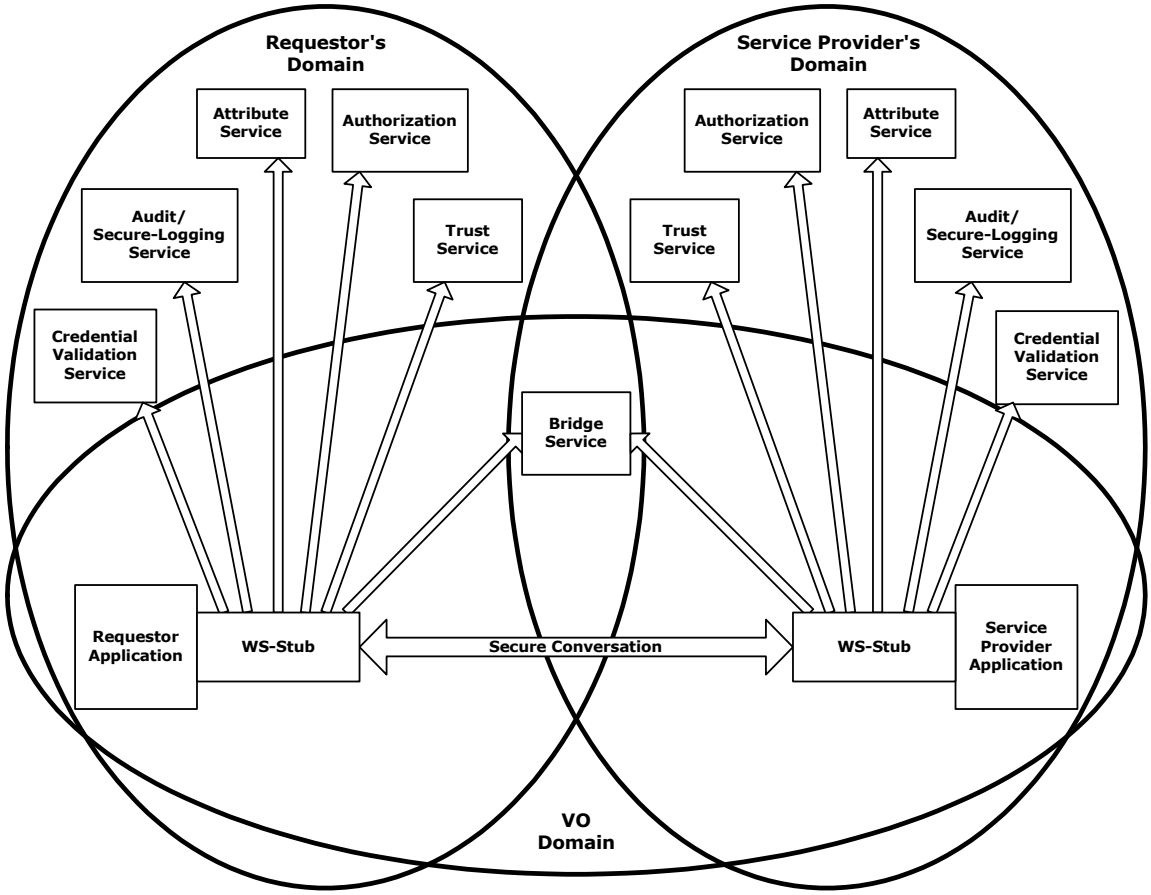


Figure: Grid Security Services

The relationship between a requestor, service provider and many of the security services is depicted in Figure in a Virtual Organization setup. All the security service portTypes used by a service requestor and service provider are to be standardized within OGSA. Compliant implementation would be able to make use of existing services and defined policies through configuration. Compliant security service implementations of a particular security related service type, would be able to provide the associated and possibly alternative security services.

7.5 Data Integration and Access

The complexity of data access and management on a grid arises from the scale, dynamism, autonomy, and distribution of data sources. These complexities should be made transparent to grid applications, through a layer of *grid data virtualization services*.

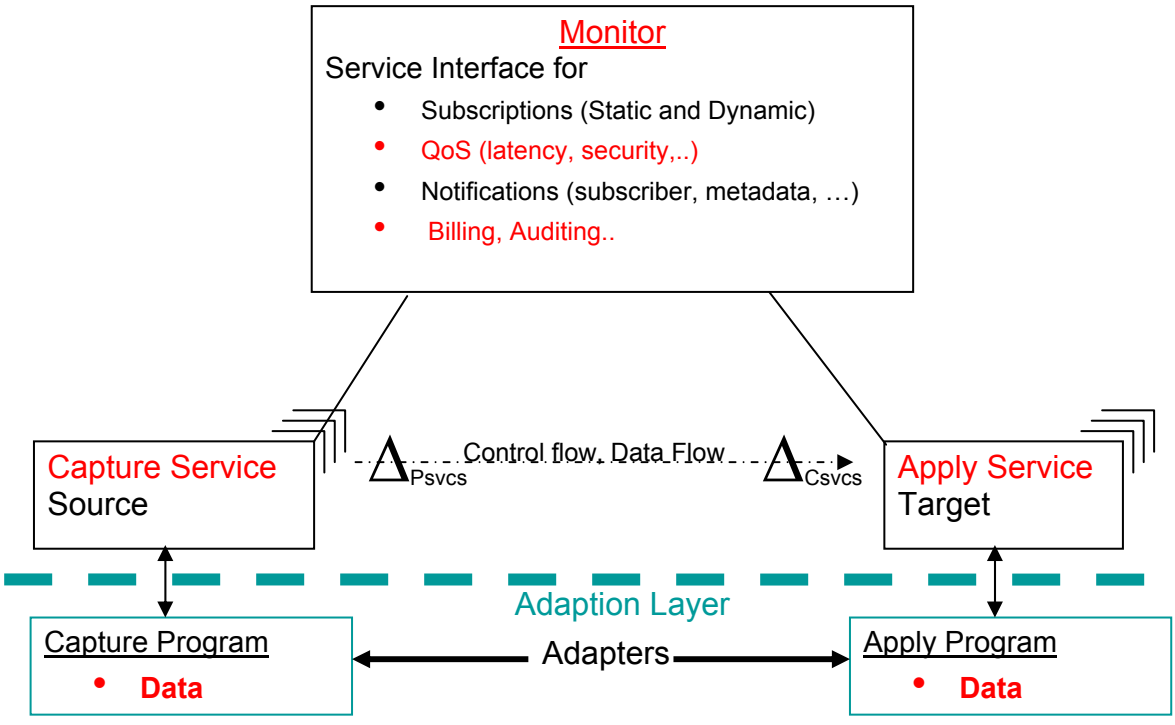
Applications require transparencies across heterogeneity, location, naming, distribution, replicas, ownership and costing for data access. The data virtualization services, such as, federated access to distributed data, dynamic discovery of data sources based on content, dynamic migration of data for workload balancing, schema management etc. provide these transparencies and enable ease of information access on a grid. The data sources need to support standard interfaces to enable uniformity of access. The GGF-DAIS group is chartered with defining the data virtualization services and the standard interfaces.

A Grid-enabled DBMS should federate all types of data - relational, XML, filesystem... It is characterized as: *Federated/Parallel DBMS which supports flexible infrastructure and dynamic data source discovery*.

7.6 Data Replication

To provide for the needs of distributed processing, the replication of data becomes critical to ensure that local compute resources have access to local-data ensuring appropriate access with satisfaction of performance objectives. Services that may consume data replication are Group Services for Clustering and failover, utility computing for dynamic resource provisioning, Policy services ensuring Quality-of-Service (QoS), Metering and Monitoring services, and also higher level Workload Management and disaster recovery solutions. Each may need to migrate data for computation or to replicate state for a given service.

The intent of the data replication work effort is to define an Open Grid Service Architecture compliant set of services that, through the use of 'adapters', can move data in and out of heterogeneous physical and logical environments without any changes needed to the underlying local data access subsystems. The adapters handle the native 'reading' and 'writing' of data and the replication software coordinates the runtime (recoverability, monitoring etc) associated with every data transfer. A central process, 'Monitor' sets up and handles communication with the calling service or program and sets up a 'subscription-pair' relationship between capture and apply services on a per-replication-request basis to ensure reliability.



7.7 Distributed Logging

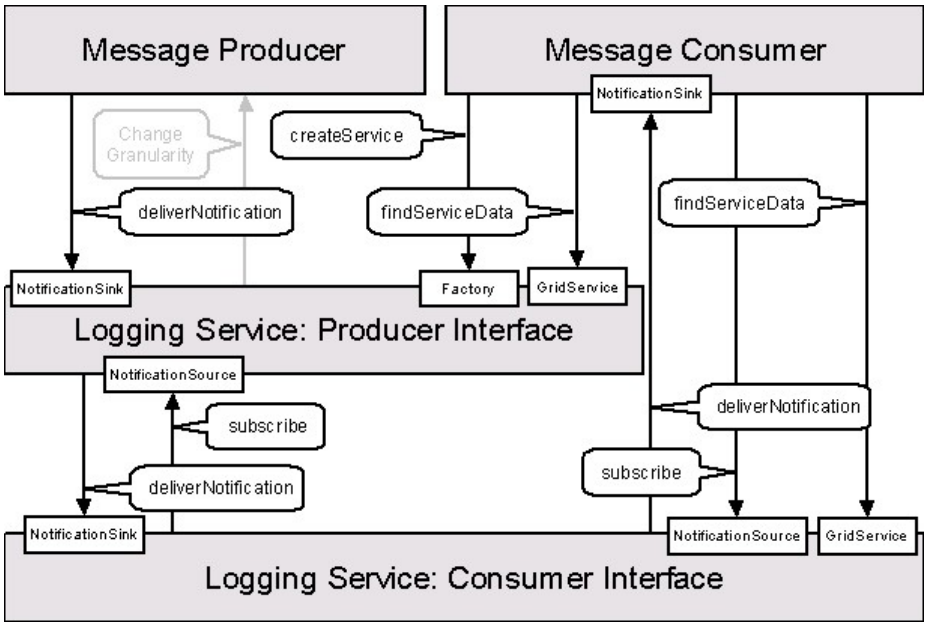
7.7.1 Motivation / Overview

One of the primary goals for OGSA is to define standard service interfaces for meta-OS functions, describe the exposed state for these functions as service data, and push down onto the native platform for implementation.

All modern operating systems and middleware provide logging systems to facilitate the communication and recording of diagnostic information. Log artifacts (i.e., atomic expressions of diagnostic information) are used by end users, system administrators, support engineers, and software developers to diagnose problems and monitor system activity. Log artifacts can also be analyzed, aggregated, and used by autonomic systems to perform corrective or preventive actions and are an essential component of Problem Determination systems.

Just as a logging service is an essential component for a traditional OS, it is also an essential meta-OS component for OGSA and every Grid Service should have access to an OGSA logging service.

Standard logging design patterns (LOG4J, JSR47, JRAS) separate the message producer component from the component that interfaces with the message consumer. The following figure expresses this basic pattern using OGSI semantics.



One of the advantages to this approach is that it permits the logical separation of logging artifact creation from logging artifact consumption; ultimate usage (e.g., logging, tracing, management) is determined by the message consumer not by the message producer.

7.7.2 Requirements

7.7.2.1.1 Abstract Interface

The log service should support standard OGSI semantics (eg FindServiceData and DeliverNotification) for accessing log artifacts. Specifically, the Logging Grid Services should be

configurable such they can be accessed through the GridService::FindSericeData operation, NotificationSink::DeliverNotification operation, or through both operations.

7.7.2.1.2 Service Data

Log artifacts should be made available as service data. Also, to facilitate standard access to log information, a canonical schema should be specified for log artifacts and its use encouraged.

7.7.2.1.3 Filtering

An application (the message producer) generates log artifacts that may or may not be used at a later time by another application (the message consumer). In most cases, the amount of data generated is very large, while the amount of data actually consumed can be relatively small. Therefore, it is desirable to have a mechanism to control the amount of data generated and to filter out data that is actually kept.

7.7.2.1.4 Configurable Behavior

Log services should support a variety of behavioral characteristics such as:

- persistency (i.e., how long should an log artifact persist, e.g., real-time monitoring data becomes irrelevant very fast, but is needed as soon as it is generated, while audit data may be needed months after it was generated),
- artifact retention policy (e.g., determining which log artifacts to drop when a buffer reaches its size limit),
- consumption of log artifacts (push/notification or pull/findServiceData),
- reliable delivery of log artifacts, and
- security (i.e., who has access to log artifacts).

Hence, there is a need for a mechanism to create different repositories of data, each with its own behavioral characteristics.

7.7.2.1.5 Native logs

To ensure access to native logs and legacy applications that exploit native logs, the logging service should be able to accommodate (surface) artifacts placed into native OS logging implementations.

7.8 Accounting: Metering resource consumption

7.8.1 Motivation / Overview

One of the primary goals for the OGSA is to define standard interfaces for meta-OS functions, describe the exposed state for these functions as service data, and push down onto the native platform for implementation.

All modern operating systems and many middleware systems provide the capability to meter resource consumption and correlate the consumption by individual users, processes, and/or threads. Traditional metering scenarios are based on departmental charge back and capacity planning applications.

An accounting service is an essential component for a traditional OS and it is an essential meta-OS component for OGSA. Traditional operating systems sample resource consumption on a per-thread or per-process basis. These resource usage statistics are then used to generate accounting records on a per user or per account basis. End-to-end flows can be constructed by composing Grid Services.

The metering of end-to-end flows in a grid environment is analogous to the metering of individual processes in a traditional OS. However, traditional OS function is not adequate for metering these end-to-end flows.

7.8.2 Requirements

An OGSA metering service must support metering the resource consumption of (configurable) classes of end-to-end flows executing on widely distributed, loosely-coupled sets of server, storage, and network resources. For example, in a departmental charge back scenario one may partition incoming requests and their subsequent flows into account classes determined by department.

In addition to traditional accounting applications, it is anticipated that end-to-end resource consumption measurements will play an important role in dynamic provisioning, and pricing grid services.

Key components of a metering grid service include modeling, obtaining (instrumentation), correlating, and logging resource usage data for resources modeled in the OGSA Common Resource Model (CRM).

7.9 Resource Management

References

1. 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.
2. Paton, N.W., Atkinson, M.P., Dialani, V., Pearson, D., Storey, T. and Watson, P. Database Access and Integration Services on the Grid, U.K. National eScience Center, 2002. www.nesc.ac.uk.
3. Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S. and Kesselman, C. Grid Service Specification, 2002. www.globus.org/research/papers/gsspec.pdf.