GWD-R (draft-ggf-ogsa-platform-2)
Open Grid Services Architecture Platform
http://www.ggf.org/ogsa-wg

Editors:
I. Foster, Argonne & U.Chicago
D. Gannon, Indiana U.

February 16, 2003

The Open Grid Services Architecture Platform

Status of this Memo

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

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 definition of a core set of interfaces, behaviors, resource models, bindings, and so forth: what we call the OGSA Platform. The OGSA working group within the Global Grid Forum has been formed to define this OGSA Platform by (a) specifying, in broad but somewhat detailed terms, the scope of important services required to support both e-science and e-business applications, (b) identifying a core set of such services that are viewed as essential for many Grid systems and applications, and (c) specifying at a high-level the functionalities required for these core services and the interrelationships among those core services. This document provides a first, and necessarily preliminary and incomplete, version of this OGSA Platform definition.

ogsa-wg@ggf.org

GLOBAL GRID FORUM
office@gridforum.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

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 within industry, e-science, or e-business. 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 Open *Grid Services Infrastructure* (OGSI) specification [3] developed within the OGSI working group of the Global Grid Forum defines, in terms of Web Services Description Language (WSDL) interfaces and associated conventions, extensions and refinements of emerging Web services standards to support basic Grid behaviors. OGSI-compliant web services—what we call Grid services—are intended to form the components of Grid infrastructure and application stacks.

OGSI defines essential building blocks for distributed systems, including standard interfaces and associated behaviors for describing and discovering service attributes, creating service instances, managing service lifetime, and subscribing to and delivering notifications. However, it certainly does not define all elements that arise when creating large-scale 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 systems in standard fashions, to achieve code reuse, and to achieve interoperability among components—three distinct and important goals.

The core set of interfaces, behaviors, profiles, models, and bindings that address these issues form what we term the *OGSA Platform.* The OGSA Platform encompasses not only broadly applicable service definitions but also models for commonly used components. The purpose of this document is to document our current understanding of the elements that must be contained within this platform, and the application requirements that motivate their inclusion. The platform definition does not address more domain-specific services, or hosting environment and protocol bindings, which we imagine will be specified in future domain- and environment-specific OGSA Platform *profile* definitions.

As we shall see, our understanding of what is required in the OGSA Platform is preliminary and incomplete. Thus, this specification should be treated as an early draft that is likely to change significantly as a result of future discussion. Both our understanding of the purpose and form of the OGSA Platform, and the details of specific components, are likely to evolve significantly. In

the meantime, this document not only provides a basis for debate but also can serve as input to discussions of priorities for OGSA specification development.

The rest of this document is structured as follows. In §2 we review briefly the scope of the OGSA-WG and the purpose of this document. In §3, we provide an overview of the use cases that we used to motivate requirements. In §4, we motivate and describe the OGSA Platform. In §5, we provide more detailed descriptions of OGSA services.

2 The OGSA Working Group

This document is a product of the Global Grid Forum's OGSA Working Group (OGSA-WG), which has the following charter and scope.

1. To produce and document the use cases that drive the definition and prioritization of OGSA Platform components, as well as document the rationale for our choices.
2. To identify and outline requirements for, and a prioritization of, OGSA Platform services.
3. To identify and outline requirements for, and a prioritization of, hosting environment and protocol bindings that are required for deployment of portable, interoperable OGSA implementations.
4. To identify and outline requirements for, and a prioritization of, models for resources and other important entities.
5. To identify, outline, and prioritize interoperability requirements for the various OGSA Platform components.
6. To define standard OGSA profiles, i.e., sets of OGSA components that meet specific requirements. E.g., the common services profile will likely include all of the shaded components in the figure; other profiles may also be defined.
7. To define relationships between GGF and other standards bodies activities such as W3C, OASIS, and WSI whose work touches upon OGSA-related issues.

In some cases, work within OGSA-WG may result in the drafting of specifications for OGSA Platform components. However, we expect that the task of completing these specifications will be handled by other working groups.

This document's preliminary definition of the OGSA Platform is intended as a contribution to goals 2, 3, and 4. It is not a final product or in any way a definite statement of the functionality required in the OGSA Platform, how this functionality should be factored into services, or how the definition of what is "standard OGSA" should be structured and presented. However, we note that we expect that this relatively simple and circumscribed "OGSA Platform" definition will be complemented by a set of "profiles" targeted to different application domains and environments.

3 Use Case Analysis

The development of this document has been informed by a variety of use case scenarios contributed by OGSA-WG participants or solicited from others. While these use cases have certainly not been defined with a view to expressing formal requirements (and do not contain the level of detail that would be required for formal requirements), they have provided useful input to the definition process. We expect to expand the number of use cases considered, as well as developing the use cases further, in future revisions of this document.

3.1 Use Cases

We expect to provide details descriptions of the use cases in a separate document; we summarize some of them briefly here to give an idea of their scope.

Commercial Data Center. This use case address the problem of managing the resources of a distributed data center on behalf of enterprises that have out-sourced the IT component of their business. The customers of the commercial Grid are the business activity mangers who are responsible for providing the specific IT services. For example, the IT business activity manager may run a "Ticketing service" that sells tickets to concert goers, or provide the database and data archival needs for a particular company.

National Fusion Collaboratory. The basic problem here is one of providing on-demand application services for large-scale data analysis and simulation. In an ideal scenario, a scientist at one of the NFC sites (a client site) needs to remotely run code installed and maintained at another NFC site (a service provider site) during an experiment within time bound.

Severe Storm Prediction. This science application involves instrument data streams from Doppler radar, satellite imaging, ground-based sensors such as pressure, temperature and humidity detectors, are constantly monitored by data mining agents looking for dangerous patterns. When one is detected, the members of a virtual organization of scientists are notified and a large number of simulations are launched automatically. Data mining tools are configured to scan the output of the simulations and compare the results against the evolving data stream from the instruments. Data archives are searched for similar patterns. Some of the instruments are automatically reconfigured to refine the data streams.

Online Media and Entertainment. Consumption of content (e.g. video on demand) does not require a lot of user interaction. Other contents, such as online games, require a lot of user interaction and it is very important to guarantee response times for these contents. A number of service providers must be integrated into a service Grid to make this work. Network service providers that offer bandwidth. A hosting capacity provider provides server and storage resource s and monitors server status and provision s additional servers. An application service providers that offer common services like online game engines, standard customer relationship management and helpdesk applications or billing applications. The content provider or studio provides the media content, artwork and game play that the consumer will experience. The integrator or publisher ties the offering together and exposes it to the consumer.

Service-Based Distributed Query Processing. The problem involves the evaluation of queries expressed in a declarative language over multiple database and computational services. The solution requires sophisticated service orchestration similar to workflow, but it also requires the ability to discover and make use of computational resources on demand, based on the anticipated resource requirements of a request.

3.2 Functionality Requirements

Analysis of the use cases just listed, other input from OGSA-WG participants, and other studies of Grid technology requirements (see Appendix) lead 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. We emphasize that this material does not represent a comprehensive or formal statement of functionality requirements from our use cases. However, it does provide useful input for subsequent development of OGSA Platform functions.

While some use cases involve highly constrained environments (that may well motivate specialized OGSA Platform profiles), it is clear that in general Grid environments tend to be heterogeneous and distributed.

- *Platforms*. The platforms themselves are heterogeneous, including a variety of operating systems (Unixes, Windows, and presumably embedded systems), hosting environments (J2EE, .NET, others), and devices (computers, instruments, sensors, storage systems, databases, networks, etc.).
- *Mechanisms*. Grid software can need to interoperate with a variety of distinct implementation mechanisms for core functions such as security.
- *Administrative environments*. Geographically distributed environments often feature varied usage, management, and administration policies (including policies applied by legislation) that need to be honored and managed.

A wide variety of application structures are encountered, and must be supported by other system components, including the following.

- Both *single-process* and *multi-process* (both local and distributed) applications covering a wide range of resource requirements.
- *Flows*, i.e., multiple interacting applications that can be treated as a single transient service instance working on behalf of a client or set of clients.
- *Workloads* comprising potentially large numbers of applications with various of the characteristics just listed.

The following basic functions are fundamental to just about every application studied.

- *Discovery and brokering.* Mechanisms are required for discovering and/or allocating services, data, and resources with desired properties.
- *Metering and accounting*. Applications and schemas for metering, auditing and billing.
- *Data sharing*. Data management and sharing are common and important Grid applications. Mechanisms are required for accessing and managing data archives, for caching data and managing its consistency, for indexing and discovering data and metadata, and so on.
- *Virtual organizations*. The need to support collaborative VOs introduces a need for mechanisms to support VO creation and management, including group membership services.
- *Monitoring*. A global, cross-organizational view of resources and assets for project and fiscal planning, troubleshooting, and other purposes.
- *Policy*. It is important to be able to represent policy at multiple stages in hierarchical systems with a view to automating the enforcement of policies that might otherwise be implemented as organizational processes or managed manually.

In addition, Grids introduce a rich set of security requirements, of which we highlight just a few here.

- *Multiple security infrastructures.* Distributed operation implies a need to interoperate with and manage multiple security infrastructures.
- *Perimeter security solutions*. Many use cases require applications to be deployed on the other side of firewalls from the intended user clients. Inter-Grid collaboration often

requires crossing institutional firewalls. OGSA needs standard, secure mechanisms that can be deployed to protect institutions while also enabling cross-firewall interaction.

Resource management is another cross-cutting requirement, encompassing service level agreement (SLA) negotiation, provisioning, and scheduling for a variety of resource types and activities.

- *Provisioning.* Computer CPUs, applications, licenses, storage, networks, and instruments are all Grid resources that require provisioning. Other new resource types will be invented and added to this list. OGSA needs a framework that allows resource provisioning to be done in a uniform, consistent manner.
- *Resource virtualization*. Dynamic provisioning implies a need for resource virtualization mechanism that allow resources to be transitioned flexibly to different tasks as required—for example, when bringing more Web servers on line as demand exceeds a threshold.
- *Optimization of resource usage* while meeting cost targets (i.e., deal with finite resources). Mechanisms to manage conflicting demands from various organizations, groups, projects and users and implementing a fair sharing of resource and access to grid.
- *Transport management.* For applications that require some form of realtime scheduling, it can be important to be able to schedule or provision bandwidth dynamically for data transfers or in support of the other data sharing applications.
- Usage models that provide for both batch and interactive access to resources.
- Support for the management and monitoring of resource usage and the detection of SLA or contract violations by all relevant parties.
- *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.

Finally, we note a number of issues that arose in multiple scenarios but that are best thought of as desirable system properties rather than functions.

- *Fault tolerance*. Support is required for fail-over, load redistribution and other techniques used to achieve fault-tolerance.
- *Disaster Recovery.* Disaster recovery is a critical capability for complex distributed Grid infrastructures. 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 should be reflected in OGSA.
- The self-healing capabilities of resources, services and systems are required. Significant manual effort should not be required to monitor, diagnose and repair faults. Ability to integrate intelligent self-aware hardware such as disks, networking devices etc.
- Strong monitoring for defects, intrusions, and other problems. Ability to migrate attacks away from critical areas.

- *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.
- *Administration*. Be able to "codify" and "automate" the normal practices used to administer the environment. The goal is that system should be able to self-organize and self-describe to manage low level configuration details based on higher-level configurations and management policies specified by administrators.

4 The OGSA Platform

As noted above, the purpose of the OGSA Platform is to define standard approaches to, and mechanisms for, basic problems that are common to a wide variety of Grid systems, such as communicating with other services, establishing identity, negotiating authorization, service discovery, error notification, and managing service collections.

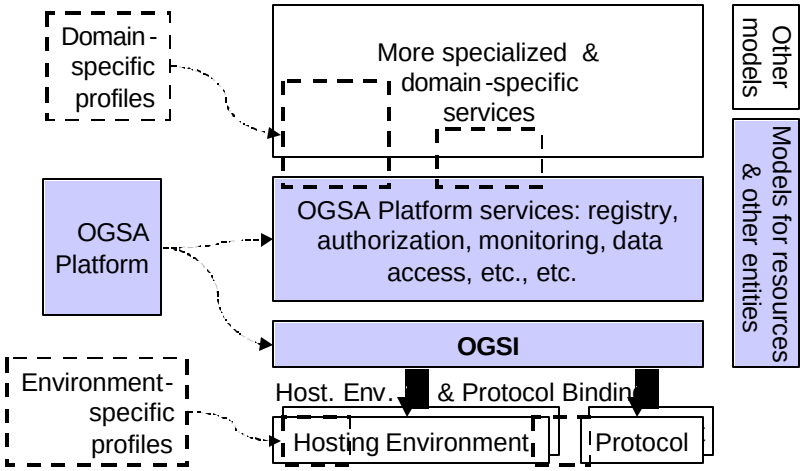

Figure 1: OGSA Platform components (shaded) and related profiles (dashed lines)

As illustrated in Figure 1, the three principal elements of the OGSA Platform are the Open Grid Services Infrastructure, OGSA Platform Interfaces, and OGSA Platform Models.

- Building on both Grid and Web services technologies, the *Open Grid Services Infrastructure* (OGSI) defines mechanisms for creating, managing, and exchanging information among entities called Grid services. Succinctly, a Grid service is a Web service that conforms to a set of conventions (interfaces and behaviors) that define how a client interacts with a Grid service. These conventions, and other OGSI mechanisms associated with Grid service creation and discovery, provide for the controlled, fault resilient, and secure management of the distributed and often long-lived state that is commonly required in distributed applications.
- *OGSA Platform Interfaces* build on OGSI mechanisms to define interfaces and associated behaviors for various functions not supported directly within OGSI, such as service discovery, data access, data integration, messaging, and monitoring.
- *OGSA Platform Models* support these interface specifications by defining *models* for common resource and service types.

We anticipate that these OGSA Platform components will be supplemented by a set of OGSA Platform Profiles addressing issues such as the following. We mention these here for completeness; they are not discussed further in this document.

- *Protocol bindings*. Profiles of this sort enable interoperability among different Grid services by defining common mechanisms for transport and authentication—issues that are not addressed by OGSI, but rather defined as binding properties, meaning that different service implementations may implement them in different ways. Thus, for example, "SOAP over HTTP" is a useful Grid service transport profile. Another example of such a profile is the recently proposed GSSAPI profile for security context establishment and message protection using WS-SecureConversation and WS-Trust [ref].
- *Hosting environment bindings*. Profiles of this sort enable portability of Grid service implementations. For example, an "OGSA J2EE Profile" might define standardized Java APIs that allow for portability of Grid services among OGSI-enabled J2EE systems. An "OGSA Desktop Grid Profile" could allow for interoperability among systems that allow untrusted (and untrusting) desktop computers to participate in distributed computations. An "OGSA Scientific Linux Profile" could define standard execution environments for computers that run scientific applications, specifying conventions for the locations of key executables and libraries, and for the names of certain environment variables
- *Sets of domain-specific services*. Profiles of this sort define interfaces and models in addition to those defined within the OGSA Platform to address the needs of specific application domains. For example, an "OGSA Database Profile" might define a set of interfaces and models for distributed database management; an "OGSA eCommerce Profile" might define interfaces and models for e-commerce applications.

We expand briefly upon each OGSA Platform elements in the remainder of this section, and provide more details in subsequent sections.

4.1 Open Grid Services Infrastructure

As noted above, the Open Grid Services Infrastructure (OGSI) defines fundamental mechanisms on which the OGSA Platform is constructed. These mechanisms address issues relating to the creation, naming, management, and exchange of information among entities called Grid services. A Grid service instance is a (potentially transient) service that conforms to a set of conventions (expressed as WSDL interfaces, extensions, and behaviors) for such purposes as lifetime management, discovery of characteristics, and notification. These conventions provide for the controlled management of the distributed and often long-lived state that is commonly required in distributed applications. OGSI also introduces standard factory and registration interfaces for creating and discovering Grid services.

4.1.1 Key OGSI Features

In the following, we introduce the key OGSI features and discuss briefly their relevance to the OGSA Platform.

Grid service descriptions and instances. OGSI introduces the twin concepts of the Grid service description and Grid service instance as organizing principles distributed systems. A Grid service description comprises the WSDL (with OGSI extensions) defining the Grid service's interfaces and service data (see next item); a Grid service instance is an addressable, potentially stateful, and potentially transient, instantiation of such a description. These concepts provide the basic building blocks used to build OGSA-based distributed systems: Grid service descriptions define interfaces and behaviors, and a distributed system comprises a set of Grid service instances that implement those behaviors, have a notion of identity with respect to the other instances in the

system, and can be characterized as state coupled with behavior published through type-specific operations.

Service state, metadata, and introspection. OGSI defines mechanisms for representing and accessing (via both queries and subscriptions) meta-data and state data from a service instance (*service data*). As well as providing a uniform mechanisms for accessing state, these mechanisms support introspection in that a client application can ask a Grid service instance to return information describing itself, such as the collection of interfaces that it implements.

Naming and name resolution. OGSI defines a two-level naming scheme for Grid service instances based on abstract, long-lived *Grid Service Handles* that can be mapped by HandleMapper services to concrete but potentially less-long-lived *Grid Service References*. These constructs are basically network-wide pointers to specific Grid service instances hosted in (potentially remote) execution environments. A client application can use a Grid Service Reference to send requests (represented by the operations defined in the interfaces of the target service) directly to the specific instance at the specified network-attached service endpoint identified by the Grid Service Reference.

Fault model. OGSI defines a common approach for conveying fault information from operations.

Lifecycle. OGSI defines mechanisms for managing the lifecycle of a Grid service instance, including both explicit destruction and soft-state lifetime management functions for Grid service instances, and Grid service factories that can be used to create instances implementing specified interfaces.

Service groups. OGSI defines a means of organizing groups of service instances.

4.1.2 Other Observations on OGSI

OGSI does not address how Grid services are created, managed, and destroyed within any particular hosting environment. Thus, services that conform to the OGSI specification are not necessarily portable across various hosting environments, but can be invoked by any client that conforms to this specification—subject, of course, to policy and compatible protocol bindings.

Stateful instances, typed interfaces, and global names are frequently also cited as fundamental characteristics of so-called distributed object-based systems. However, various other aspects of distributed object models (as traditionally defined) are specifically not required or prescribed by OGSI. For this reason, we do not adopt the term distributed object model or distributed object system when describing this work, but instead use the term Open Grid Services Infrastructure, thus emphasizing the connections with both Web services and Grid technologies.

Among the object-related issues that are not addressed within OGSI are implementation inheritance, service mobility, development approach, and hosting technology. The Grid service specification does not require, nor does it prevent, implementations based upon object technologies that support inheritance at either the interface or the implementation level. There is no requirement in the architecture to expose the notion of implementation inheritance either at the client side or the service provider side of the usage contract. In addition, the Grid service specification does not prescribe, dictate, or prevent the use of any particular development approach or hosting technology for the Grid service. For example, there is nothing about OGSI that is Java-specific: one can implement OGSI behaviors in C, Python, or other languages.

Grid service providers are free to implement the semantic contract of the service in any technology and hosting architecture of their choosing. We envision implementations in J2EE, .NET, traditional commercial transaction management servers, traditional procedural UNIX servers, etc. We also envision service implementations in a variety of programming languages that would include both object-oriented and non-object-oriented *alternatives.*

4.2 OGSA Platform Interfaces

The OGSA Platform Interfaces (and associated behaviors) define functions that occur within a wide variety of Grid systems. We divide these functions into

Name resolution and discovery. OGSI's two-level name space requires HandleResolver services capable of resolving from Grid service handles (GSHs) to Grid service references (GSRs). The OGSA Platform should define one or more standard behaviors for, and/or specializations of, the OGSI HandleResolver interface, to permit GSH resolution in various settings. The OGSA Platform should also define standard service registration and discovery interfaces for maintaining and querying mappings from semantic information (e.g., keywords or directory structures) to GSHs. These latter interfaces can build on OGSI ServiceGroup mechanisms.

Service domains. It appears likely to be common practice for an OGSA-compliant "service" to be implemented via a collection of internal services that are managed in some coordinated fashion. OGSA Platform service domain interfaces and behaviors facilitate the creation and operation of, and the integration of new services into, such *service domains*.

Security. This category is wide-reaching, encompassing issues relating to the management and verification of credentials; privacy and integrity; and policy (discussed separately below). Requirements here are wide reaching, encompassing policy services. 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.

Policy. A policy is a definitive goal or 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 security, workload management, and qualities of service, and provide a set of rules to administer, manage and control access to Grid resources. OGSA Platform policy interfaces provide a framework for creating, managing, validating, distributing, transforming, resolving, and enforcing policies within a distributed environment.

Data management. 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.

Messaging, queuing, and logging. Messaging and Queueing services that are exposed as a set of core services of OGSA. This is a collection of services that includes pub-sub, topic based messaging and queue based message. Job and task queueing is also supported by these abstractions. A logging service is a form of mediator between message producer and message consumer that can provide archival storage for message streams. Defining standard interfaces to logging services is important as it allows different clients to exploit the same common logging service implementations.

Events. An event is a representation of an occurrence in a system or application component that may be of interest to other parties. Standard means of representing, communicating, transforming, reconciling, and recording events are important for interoperability.

Metering and accounting. Accounting mechanisms are used to collect, deliver, reconcile, store securely, and manage information about resource usage and/or charges. This may be a higher order service. Accounting often involves extracting information that may come from a logging service to determine rating billing. Accounting schemas are also needed.

Transactions. Transaction interfaces encapsulate protocols used to assure all parties that transactions have executed correctly in a distributed environment.

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.

4.3 Common Models

OGSI service data and associated operations provide basic machinery for monitoring and managing Grid service instances. One just needs to define appropriate service data (and policies for governing who can access that service data) and then link this service data to appropriate service state; authorized clients can then query that service data, request notifications when it is modified, and/or change the service data's value. Underlying monitoring and management functions can be implemented via application-specific mechanisms, or via standards such as CIM, SNMP, LDAP, etc.

These mechanisms introduces the need for standard schema, and a variety of such standard schema will presumably form part of the OGSA Platform.

One area in which discussion has already started concerns *common resource models*, a term used to denote an abstract representation of an IT Resource, such as node, interface adaptor, disk, filesystem, or IP address. Such a model can map directly to a physical resource, or alternatively serve as an abstract representation of a logical resource constructed from multiple physical resources to build higher-level services and 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 GridService interface plus additional interfaces 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 GridService interface's find and query operations. Additional interfaces provide manageability interfaces 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 state transitions that a resource (in CRM, a service) goes through. Resources exist from the time they are installed until they are destroyed, and can be (and in most cases, are) managed in different ways over their lifetime. The resource lifecycle extensions describe the meaningful lifecycle states and transitions for the service, i.e., interfaces, 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.

5 OGSA Platform Interfaces

In this section we provide a more detailed view of the OGSA Platform interfaces introduced in §4.2.

5.1 Service Groups and Discovery Interfaces

Grid Service Handles (GSHs) and Grid Service References (GSRs) together realize a two-level naming scheme, with HandleResolver services mapping from handles to references. However, GSHs are not intended to contain semantic information and indeed may be viewed for most purposes as opaque. Thus other entities (both humans and applications) need other means for discovering services with particular properties, whether relating to interface, function, availability, location, policy, or other criteria.

Traditionally in distributed systems this problem is addressed by creating a third-level "human-readable" or "semantic" name space that is then mapped (bound) to abstract names (in our case, GSHs) via registry, discovery, metadata catalog, or other similar services. It is important that the OGSA Platform define standard functions for managing such name spaces, as otherwise services and clients developed by different groups cannot easily discover each other's existence and properties. These functions must address the creation, maintenance, and querying of name mappings. Two types of such semantic name spaces are common—naming by attribute, and naming by path.

5.1.1 Attribute Naming and Registries

Attribute naming schemes associate various metadata with services and support retrieval via queries on attribute values. A registry implementing such a scheme allows service providers to publish the existence and properties of the services that they provide, so that service consumers can discover them. We envision special purpose registries being built on the base service group mechanisms provided by the OGSI definition. In other words, an OGSA-compliant registry is a concrete specialization of the OGSI service group.

A ServiceGroup is a collection of entries, where each entry is a Grid service implementing the ServiceGroupEntry interface. The ServiceGroup interface also extends the GridService interface. There is a ServiceGroupEntry for each service in the group (i.e., for each group member). Each ServiceGroupEntry contains a serviceLocator for the referred-to service and information (Content) about that service. The content element is an XML element advertising some information about the member service. The type of the Content element conforms to one of the QName elements in the ContentModelType SDE of the ServiceGroup interface.

It is the content model of the service group definition that suggests the concrete type and specific use of the registry being offered. The content model of the serviceGroupEntry for a given service group is published in the service data of the service group. The content model is the basis on which search predicates can be formed and executed against the service group with the findServiceData operation. In other words, it is the content model that forms the basis of the

registry index upon which registry searches can be executed. It is envisioned that many application specific, special purpose registries will be developed.

It is also envisioned that many registries will inherit and implement the notificationSource interface so as to facilitate client subscription to register state changes. Again, specific state change subscriptions will be possible through the advertisement of the registry specific service group content model of the service group on which the registry is built.

As stated earlier, we envision many application specific registry implementations being defined. Whether or not one or more general purpose registry types should be defined and adopted as part of the OGSA Platform is to be determined.

5.1.2 Path Naming and Directories

Path naming or *directory* schemes (as used, for example, in file systems) represent an alternative approach to attribute schemes for organizing services into a hierarchical name space that can be navigated. The two approaches can be combined, as in LDAP. Directory path naming can be accomplished by defining a PathName Interface that maps strings to GSHs. Thus a string such as "/data/genomics_dbs/mouse" could map to a service (GSH) that might deliver portions of the mouse genome, and perhaps also do BLAST searches against the mouse genome. Similarly, "/applications/biology/genomics/BLAST" could map to a GSH that has Interfaces for executing BLAST.

The Interface will have methods to insert, lookup, and delete <string, GSH> pairs, and will in essence be a simple table. It is expected that path_name services will be "chained" together, so that evaluation of a path may involve traversing several path_name services, forming a directed graph. This can be used to link disjoint namespaces into namespace cliques.

5.2 Service Domain Interfaces

The value of Grid solutions will be realized through the formation of Grid service collections and the orchestration of automated interactions among services and across collections. In addition to identifying specific common services, the OGSA Platform must describe the common behaviors, attributes, operations and interfaces needed to allow services to interact with others in a fully distributed, heterogeneous, but Grid-enabled environment; and for a collection of underlying services to be composed (perhaps recursively) into high-order services as an integral unit to serve a domain-specific functional purpose—what we call here a *Service Domain*. The latter functionality introduces a need to support the registration, discovery, selection, filtering, routing, fail-over, creation, destroying, enumeration, iteration, and topological mapping of service instances represented by a service domain collection, as well as intra and inter collection interactions.

In addressing these requirements, we have as a building block the OGSI service group interfaces, which define the abilities to register (add) and unregister (remove) service instances from a set called a service group. The OGSA Platform should extend these interfaces to provide a rich set of behaviors (and associated operations and attributes) for service domain management. The following are candidates.

- *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 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 within the service domains.

5.3 Security

OGSA Platform security mechanism 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 the Global Grid Forum'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 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, as depicted in Figure 2 and discussed in the following.

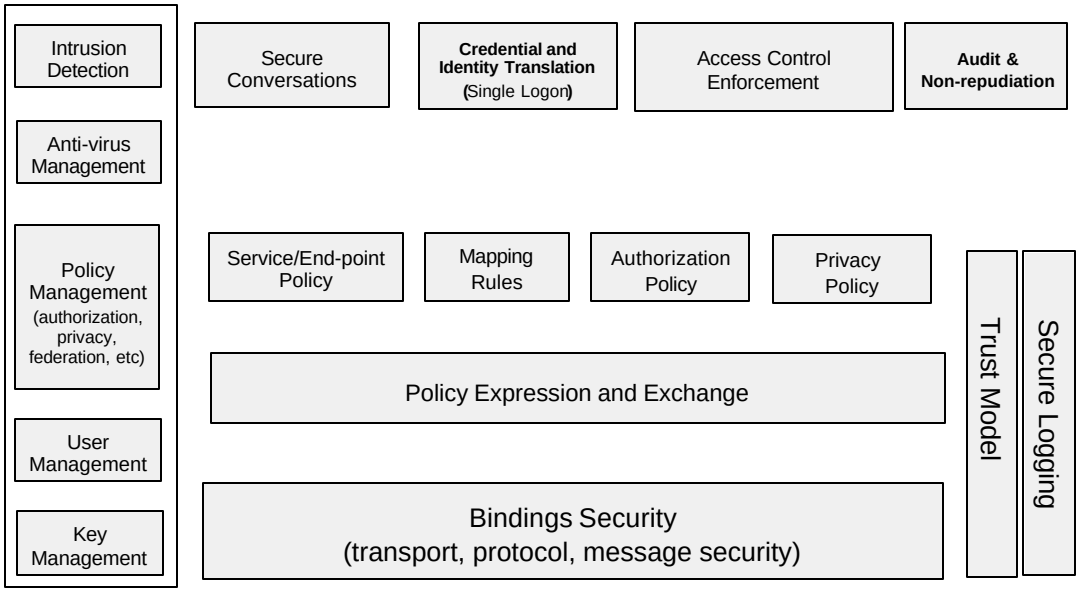

Figure 2: Schematic of a proposed OGSA security architecture

As discussed in §4, a Grid service can be accessed over a variety of protocol bindings. Given that bindings deal with protocol and message formats, security functions as confidentiality, integrity, and authentication fall within the scope of bindings and thus are outside the scope of the OGSA Platform proper—but not specific OGSA Platform profiles.

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. When invoking Grid services dynamically, end points may need to discover the policies of a target service and establish trust relationships dynamically. (See §5.4 for more discussion of policy.)

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.

Thus the OGSA Platform's 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. We can expect that existing and evolving standards will be adopted or recognized in the Grid security model.

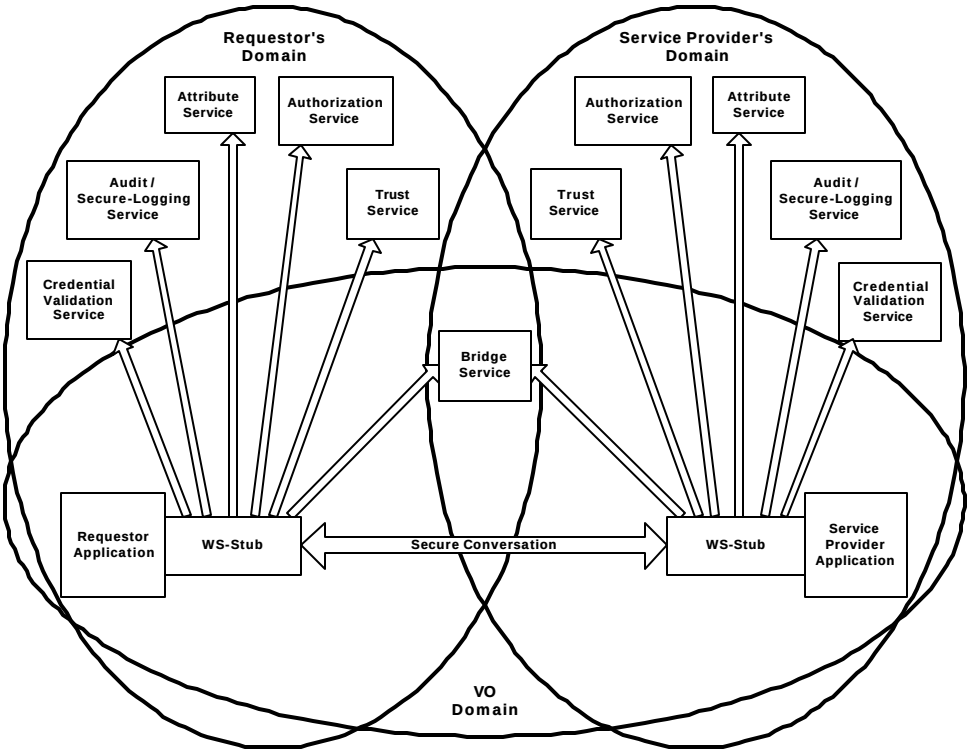

Figure 3: Security services in a virtual organization setting

The relationship between a requestor, service provider and many of the security services is depicted in **Error! Reference source not found.** in a Virtual Organization setup. All security interfaces used by a service requestor 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.

5.4 Policy

We can expect that many Grid services will use policies to direct their actions. Thus, Grids need to support the definition, discovery, communication, and enforcement of policies for such purposes as resource allocation, workload management, security, automation, and qualities of services. Some policies need to be expressed at the operational level, i.e., at the level of the devices and resources to be managed, while higher-level policies express business goals and service level agreements (SLA) within and across administrative domains. Higher-level policies are hard to enforce without a canonical representation for their meaning to lower-level resources.

Thus, business polices probably need to be translated into a canonical form that can then be used to derive lower-level policies that resources can understand. Standard mechanisms are also needed for managing and distributing policies from producers (e.g., administrators, autonomic managers, SLAs, etc.) to end-points that consume and enforce them (i.e., devices and resources).

To meet these requirements, the OGSA Platform needs to define the representations and functions required to implement an end-to-end distributed policy management service. These representations and functions are likely to include the following.

- A canonical representation for expressing policies (Policy Information Model and Core XML Schema)
- A management control point for policy lifecycle (Policy Service Manager interface)
- An interface that policy consumers can use to retrieve required polices (Policy Service Agent interface)
- A way to express that a service is "policy aware" (Policy Enforcement Point interface)
- A way to effect change on a resource (e.g., using Common Resource Models: §4.3)

These interfaces provides a framework for creating, managing, validating, distributing, transforming, resolving, and enforcing policies within a distributed environment. The Policy Service Manager controls access to the policy repository. It also controls when notifications of policy changes are send out so that multiple updates can be made and notifications are only send after all updates are complete. The Policy Service Agent is the service that "policy aware" services go to for their policies. The agent can provide additional services like understanding time-period conditions so it can inform policy consumers of when policies become active or inactive. Services that consume policies will implement the Policy Enforcement Point interface to allow them to be registered with Policy Agents, participate in the subscription to and notification of policy changes, and to allow policies to be pushed down onto them when needed. These enforcement points will need to interpret the policies and make the necessary configurations changes in the resource they manage, by using the Common Resource Model mechanisms referred to in §4.3. The OGSA Policy Service provides for a transformation service to fill this purpose and includes a canonical representation of policy in the form of an information model, grammar, and core XML schema.

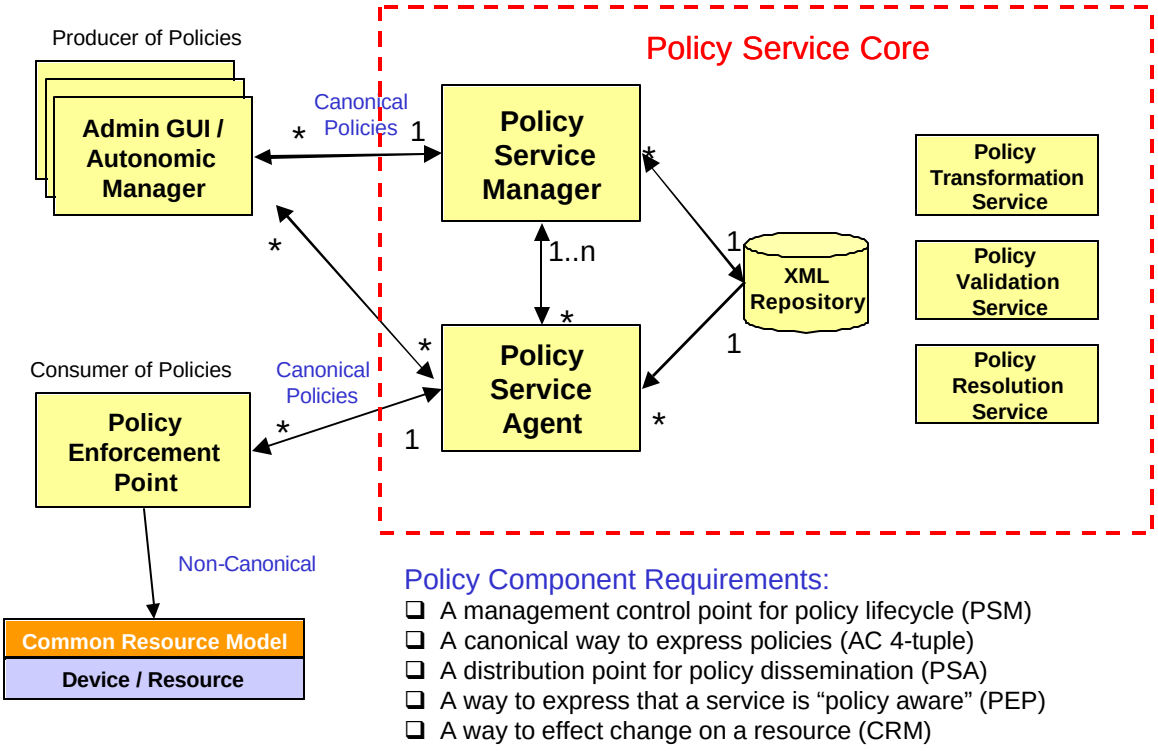

Figure 4: A set of potential policy service components

A set of secondary validation interfaces can allow automated managers and administrators to act on the same set of policies and validate consistency. An interface is also required for translating policies to and from the canonical form so that consumers that have their own policy formats can plug into the service. Finally there is a need for run-time resolution of policy conflicts, which may require specific application knowledge to determine the cost of violating an agreement and selecting the policy that that will have appropriate impact.

5.5 Data Management Services

The scale, dynamism, autonomy, and distribution of data sources in Grid environments can lead to significant complexity in data access and management. A variety of interfaces need to be defined to aid developers and users in the management of this complexity. In addition to basic data access interfaces and common resource models for storage and data management systems, these interfaces need to address the need for various transparencies, including heterogeneity, location, naming, distribution, replicas, ownership, and data access costs. Data virtualization services aimed at providing these transparencies can include federated access to distributed data, dynamic discovery of data sources based on content, dynamic migration of data for workload balancing, and schema management. In implementing such services, we need to take into account the wide variety of different data types, such as flat file data, streaming media, and relational data that require different approaches to management. Further, different applications require different forms of support, e.g., some applications cannot be modified and require transparent access via file systems, while others need explicit management of data locality and replication.

These considerations suggest a role for wide variety of potential data management interfaces, including data caching (resolving a file handle to a flat file into a data stream); data replication; data access, via mechanisms for accessing wide range of data types, including flat files, RDBMS, and streaming media; file and DBMS services and possibly federated data management services that are used as part of a vertical utility Grid; data transformation and filtering; schema

transformation (allowing different data, service and policy schema to be reconciled so that the services can interact correctly); and Grid storage services, which allow direct access to storage throughout the Grid.

We do not yet know which, if any, of these various interfaces should be viewed as sufficiently fundamental to justify inclusion in the OGSA Platform. However, we provide some material on requirements for data management in general.

Data access services. Basic data access interfaces allow clients to directly access and manipulate data. A number of such interfaces are required, corresponding to different data types, e.g., files, directories, file systems, RDBMS, XML data bases, object data bases, and streaming media. A "file access" service may export interfaces to *read*, *write*, *truncate*. GridFTP, an existing data access service, provides mechanism to *get* and *put* files, and supports third party transfers.

Data replication. Data replication can be important as a means of meeting performance objectives by allowing local compute resources to have access to local data. While closely related to caching (indeed, a "replica store" and a "cache" may differ only in their policies), replicas may provide different interfaces. Services that may consume data replication are group services for clustering and fail-over, utility computing for dynamic resource provisioning, policy services ensuring various qualities of service, 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.

Work is required to define an OGSA-compliant set of data replication 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 "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.

Data caching services. In order to improve performance of access to remote data items caching services will be employed. At the minimum caching services for traditional flat file data will be employed. Caching of other data types, such as views on RDBMS data, streaming data, and application binaries are also envisioned. Issues that arise include (but are not limited to):

- Consistency – Is the data in the cache the same as in the source? If not, what is the coherence window? Different applications have very different requirements.
- Cache invalidation protocols – How and when is cached data invalidated?
- Write through or write back? When are writes to the cache committed back to the original data source?
- Security – How will access control to cached items be handled? Will access control enforcement be delegated to the cache, or will access control be somehow enforced by the original data source?
- Integrity of cached data – Is the cached data kept in memory or on disk? How is it protected from un-authorized access? Is it encrypted?

How the cache service addresses these issues will need to available as service data.

Metadata catalog and service services. These services allow us to search for GSHs or data directory entries based on object metadata attributes. 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. They are also closely related to registry services.

Schema transformation. Schema transformation interfaces support the transformation of data from one schema to another. For example, XML transformations as specified in XSLT.

Storage. Storage can be modeled as a service just as any other resource. Grid storage interfaces can be represented as CRM services.

5.6 Messaging and Queuing

OGSA extends the scope of the base OGSI Notification interface to allow Grid services to produce a range of event messages – not just notifications that a serviceData element has changed.

Several terms related to this work are:

- Event - Some occurrence within the state of the Grid Service or its environment that may be of interest to third parties. This could be a state change or could be environmental, such as a timer event.
- Message - An artifact of an event, containing information about an event that some entity wishes to communicate to other entities
- Topic - A "logical" communications channel and matching mechanism to which a requestor may subscribe to receive asynchronous messages and publishers may publish messages.

A message is represented as an XML element with a namespace-qualified QName, and an XML Schema-defined complex type. A Topic will be modeled as an XML element, describing its internal details, including expected messages associated with the topic. TopicSpaces, or collections of Topics will also be modeled.

This work will also define:

- An interface to allow any Grid service to declare its ability to accept subscriptions to topics and the topics its supports.
- An interface to describe a messaging intermediary (a message broker) that supports anonymous publication and subscription on topics.
- An interface (or set of interfaces) that describe the interface to other messaging services such as a Queuing service.

Note that queuing and message qualities of service such as reliability can be considered both an explicit service within an OGSA hosting environment and a transport detail modeled by the wsdl:binding element in the service description.

5.7 Events

An event is a representation of an occurrence in a system or application component that may be of interest to other parties. Standard means of representing, communicating, transforming, reconciling, and recording events are important for interoperability. Thus the OGSA Core should define:

- Standard schema seem desirable for at least certain classes of OGSA events. Topics to be addressed include: Is there an "OGSA Event base class"? Is there standard content for events, such as source, name, and details? We note that there is an OASIS group working

on standard schema for Web service events (the OASIS Management Protocol TC). Is this group addressing what we need, or are there unique OGSA requirements?

- Standard interface(s) for communicating events with specified QoS. These may be based directly on the Messaging interfaces.
- Standard interface(s) for transforming (mediating) events in a manner that is transparent to the endpoints.
- Standard interface(s) for reconciling events from multiple sources.
- Standard interface(s) for recording events. These may be based directly on the Message logging interface(s).

Note: Event services applied to fault tolerance.

5.8 Distributed Logging

Distributed logging can be viewed as a typical messaging application in which *message producers* generate *log artifacts*, i.e., atomic expressions of diagnostic information, that may or may not be used at a later time by other, independent, *message consumers*. OGSA-based logging can leverage the notification mechanism available in OGSI as the transport for messages. However, it is desirable to move logging-specific functionality to intermediaries, or *logging services*. Such logging services provide the extensions needed to deal with the following issues.

- *Decoupling*: The logical separation of logging artifact creation from logging artifact consumption. The ultimate usage of the data (e.g., logging, tracing, management) is determined by the message consumer; the message producer should not be concerned with this.
- *Transformation and common representation*: Logging packages commonly annotate the data that they generate with useful common information such as category, priority, timestamp, and location. An OGSA logging service should not only provide the capability of annotating data, but also the capability of converting data from a range of (legacy) log formats into a common standard canonical representation. Also, a general mechanism for transformation may be required (based on XSLT).
- *Filtering and aggregation*: The amount of logging data generated can be large, while the amount of data actually consumed can be small. Therefore, it can be desirable to have a mechanism for controlling the amount of data generated and for filtering out what is actually kept and where. Through the use of different filters, data coming from a single source can be easily separated into different repositories, and/or "similar" data coming from different sources can be aggregated into a single repository.
- *Configurable persistency*: Depending on consumer needs, data may have different durability characteristics. For example, in a real-time monitoring application, data may become irrelevant quickly, but is needed as soon as it is generated; data for an auditing program may be needed months or even years after it was generated. Hence, there is a need for a mechanism to create different data repositories, each with its own persistency characteristics. In addition, the artifact retention policy (e.g., determining which log artifacts to drop when a buffer reaches its size limit) should be configurable.
- *Consumption patterns*: Consumption patterns differ according to the needs of the consumer application, for example, a real time monitoring application needs to be notified whenever a particular event occurs, while a post-mortem problem determination program queries historical data trying to find known patterns. Thus, the logging

repository should support both synchronous query- (pull-) based consumption and asynchronous push-based (event-driven) notifications. The system should be flexible enough that consumers can easily customize the event mechanism—for example, by sending digests of messages instead of each one—and maybe even provide some predicate logic on log artifacts to drive the notifications.

These considerations lead us to define an architecture for OGSA logging services (Figure XX) in which producers talk to *filtering and transformation* services either directly, or indirectly through adapters. Consumers also use this service to create custom message repositories (*baskets*) or look for existing producers and basket, i.e., this service should also function as a factory (of basket) and a registry (of producers and baskets). There is also a need for a configurable *storage and delivery* service, where data from different filtering services is collected, stored, and, if required, delivered to interested consumers.

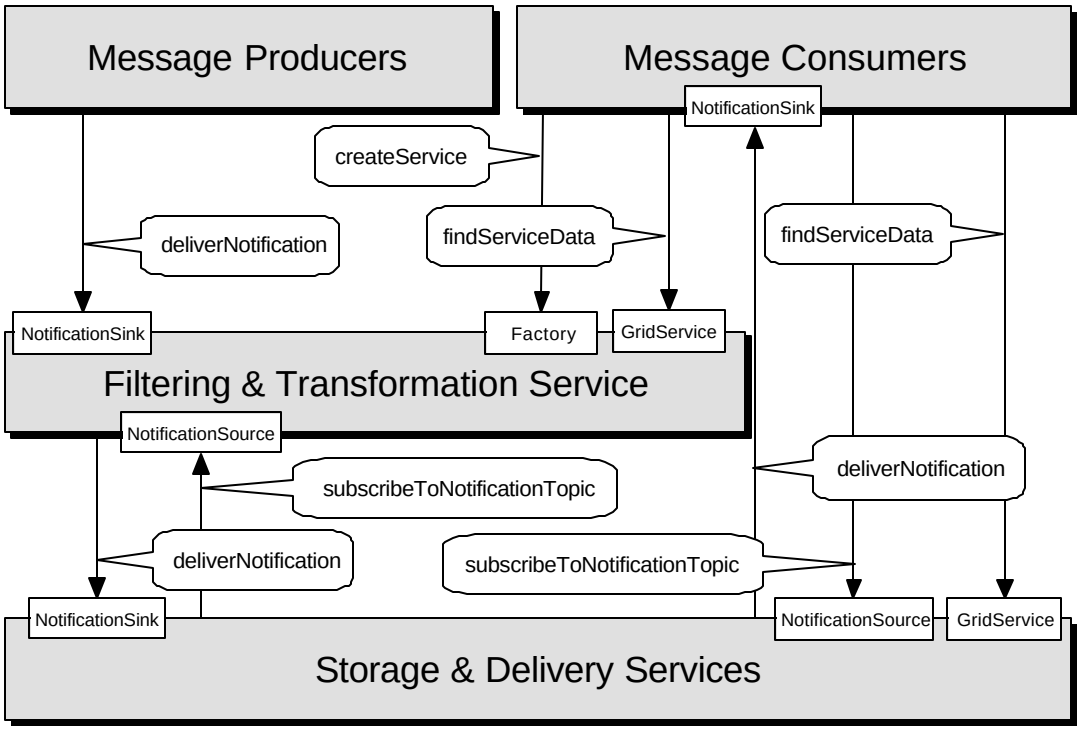

Figure 5: Schematic of a messaging service architecture

5.9 Metering and Accounting

Different Grid deployments may integrate different services and resources and feature different underlying economic motivations and models. However, regardless of these differences, it is a quasi-universal requirement that resource utilization can be monitored, whether for purposes of cost allocation (i.e., charge-back), capacity and trend analysis, dynamic provisioning, grid-service pricing, fraud and intrusion detection, and/or billing. OGSA Platform metering and accounting interfaces address this requirement by defining standard monitoring, metering, rating, accounting, and billing interfaces.

We expect a close relationship between the interfaces discussed here and the proposed Common Resource Model (CRM). CRM can provide access to basic resource performance and utilization instrumentation, exposed as serviceData. For example, an operating system might publish counter values corresponding to the state of system activities such as CPU utilization, buffer usage, disk

and tape I/O activity, TTY device activity, switching and system-call activity, file-access, queue activity, interprocess communications, and paging—all metrics that may be useful for purposes of metering and accounting.

5.9.1 Metering Interface

A Grid service may consume multiple resources and a resource may be shared by multiple service instances. Ultimately, the sharing of underlying resources is managed by middleware and operating systems. All modern operating systems and many middleware systems have metering sub-systems for measuring resource consumption (i.e., monitored data) and for aggregating the results of those measurements. For example, all commercial Unix systems have provisions for aggregating prime time and non-prime time resource consumption by user and command.

A metering interface provides access to a standard description of such aggregated data (metering serviceData). A key parameter is the time window over which measurements are aggregated. In commercial Unix systems, measurements are aggregated at administrator-defined intervals (cron entry), usually daily, primarily for the purpose of accounting. On the other hand, metering systems that drive active workload management systems might aggregate measurements using time windows measured in seconds. Dynamic provisioning systems use time windows somewhere between these two examples.

Several use cases require metering systems that support multi-tier, end-to-end flows involving multiple services. An OGSA metering service must be able to meter the resource consumption of configurable classes of these types of flows executing on widely distributed, loosely coupled server, storage, and network resources. Configurable classes should support, for example, a departmental charge back scenario where incoming requests and their subsequent flows are partitioned into account classes determined by the department providing the service. The metering of end-to-end flows in a grid environment is somewhat analogous to the metering of individual processes in a traditional OS. Since traditional middleware and operating systems do not support this type of metering, additional function must be accommodated by OGSA. 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.

Finally, in addition to metering resource consumption, metering systems must also accommodate the measurement and aggregation of application-related (e.g., licensed) resources. For example, a grid service might charge consuming services a per-use fee. The metering service must be able to support the measurement of this class of service (resource) consumption.

5.9.2 Rating Interface

A rating interface needs to address two types of behaviors. First of all, once the metered information is available, it has to be translated into financial terms. That is, for each unit of usage, a price has to be associated with it. This step is accomplished by the rating interfaces, which provides operations that take the metered information and a rating package as input and output the usage in terms of chargeable amounts. For example, a commercial UNIX system indicates that 10 hours of prime-time resource and 10 hours on non-prime-time resource are consumed, and the rating package indicates that each hour of prime-time resource is priced at 2 dollars and each hour of non-prime-time resource is priced at 1 dollar, a rating service will apply the pricing indicated in the rating package and translate the usage information into financial information in the terms of 20 dollars of prime-time resource charge, and 10 dollars of non-prime time resource charge.

Secondly, when a business service is developed, a rating service is used to aggregate the costs of the components used to deliver the service, so that the service owner can determine the pricing, terms and conditions under which the service will be offered to subscribers.

5.9.3 Accounting Interface

Once the rated financial information is available, an accounting service can manage subscription users and accounts information, calculate the relevant monthly charges and maintain the invoice information. This service can also generate and present invoices to the user. Account-specific information is also applied at this time. For example, if a user has a special offer of 20% discount for his usage of the commercial UNIX system described above, this discount will be applied by the accounting service to indicate a final invoiced amount of 24 dollars.

5.9.4 Billing/Payment Interface

Billing/Payment service refers to the financial service that actually carries out the transfer of money. For example, a credit card authorization service.

5.10 Administrative Services

Administrative services automate or otherwise assist with a variety of installation, maintenance, monitoring, and troubleshooting tasks within a Grid system. For example, system administrators today can face the task of installing hundreds of components within an operational data center. Complex and sometimes circular dependency relationships between different components can make this installation process tedious and time consuming. One approach to automating this installation process in a generic fashion would be to define standard data schema for describing installation dependencies (e.g., service A requires a particular quality of service from service B), standard data schema describing steps of installation, and data-driven services that trigger installation and configuration actions.

We are not ready to define requirements for OGSA Platform interfaces in this area, but anticipate defining them in the future.

5.11 Transactions

Transaction services are important in many Grid applications, particularly in industries such as financial services and in application domains such as supply chain management. However, transaction management in a widely distributed, high latency, heterogeneous RDBMS environment is more complicated than in a single machine room with a single vendor's software. Traditional distributed transaction algorithms, such as two-phase distributed commit, may be too expensive in a wide area grid, and other techniques such as optimistic protocols may be more appropriate. At the same time, different applications often have different characteristics and requirements that can be exploited when selecting a transaction technique to use. Thus, it is unlikely that there will be a "one size fits all" solution to the transaction problem.

Transaction services are also being closely examined in the Web services community. For example, WS-Transactions has been recently "proposed." This initiative should be closely tracked.

5.12 Grid Service Orchestration

Grid Service orchestration refers to the problem of describing and managing the choreography of a set of interacting services, perhaps on multiple distributed resources. This problem arises in many setting and it seems desirable to define standard Grid Service orchestration interfaces for such basic activities as defining a workflow, monitoring the execution of a workflow, and editing

or otherwise managing the execution of the workflow. We do not yet know in detail what form these interfaces should take, so just make a few general comments here.

Rather than assuming a "workflow language standard" (a goal perhaps as unadvisable as specifying a "standard programming language"), GSO interfaces can provide a standard port type for launching an instance of an orchestration task. Different Grid workflow engine factories may implement or extend this interface. If each such service is registered as a member of a Grid Service orchestration service group, a client can select the appropriate service based on the specific orchestration task or language required. The orchestration service mediates interactions among sub-services and handles the exceptions and faults that may occur in the orchestration execution.

Additional Grid Service orchestration interfaces would be associated with, and implemented by, each instance of a orchestration task. Those interfaces allow clients to register for notification about the progress of orchestration or to directly request state information that is specific to that orchestration mechanism.

6 Security Considerations

This specification defines requirements for interfaces, behaviors, and models used to structure and achieve interactions among Grid services and their clients. While it is assumed that such interactions must be secured, the details of security are out of 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.

7 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

Dennis Gannon
Indiana University
Bloomington, IN xxxxx
gannon@cs.indiana.edu

8 Contributors

We gratefully acknowledge the contributions made to this specification by Takuya Araki, Jamie Bernardin, Shel Finkelstein, Jeffrey Frey, Andrew Grimshaw, Kate Keahey, Hiro Kishimoto, Tan Lu, Fred Maciel, David Martin, Jeffrey Nick, David Snelling, Ravi Subramaniam, and Jay Unger.

9 Acknowledgements

This work was supported in part by IBM and by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38 and DE-AC03-

76SF0098; by the National Science Foundation; and by the NASA Information Power Grid project.

Appendix: Information Sources

In identifying services we draw upon the following sources (references to be provided):

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

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.