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

September 21, 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.

ITF: Do we want to talk about the "OGSA Platform" or simply OGSA? I can't recall the outcome of our earlier discussion on this topic. I myself think that in practice we have NOT been able to identify any universally "core services" beyond OGSI; thus, this terminology is wrong, and the document should just be "Open Grid Services Architecture"

ogsa-wg@ggf.org

Full Copyright Notice

Intellectual Property Statement

Contents

Deleted: 8
Deleted: 8
Deleted: 11
Deleted: 12
Deleted: 13
Deleted: 13
Deleted: 14
Deleted: 14
Deleted: 15
Deleted: 16
Deleted: 17
Deleted: 18
Deleted: 19
Deleted: 19
Deleted: 19
Deleted: 20
Deleted: 21
Deleted: 21
Deleted: 21
Deleted: 21
Deleted: 21
Deleted: 21
Deleted: 21
Deleted: 21
Deleted: 21
Deleted: 21
Deleted: 22
Deleted: 23
Deleted: 23
Deleted: 23
Deleted: 23
Deleted: 24
Deleted: 24
Deleted: 24
Deleted: 24
Deleted: 24
Deleted: 24
Deleted: 24
Deleted: 24
Deleted: 24
Deleted: 25
Deleted: 25

Deleted: 26
Deleted: 26
Deleted: 26
Deleted: 26
Deleted: 26
Deleted: 26
Deleted: 26
Deleted: 26
Deleted: 27
Deleted: 27
Deleted: 27
Deleted: 27
Deleted: 27
Deleted: 27
Deleted: 27
Deleted: 27
Deleted: 27
Deleted: 27
Deleted: 27
Deleted: 29
Deleted: 29
Deleted: 29
Deleted: 29
Deleted: 29
Deleted: 29
Deleted: 29
Deleted: 29
Deleted: 29
Deleted: 30
Deleted: 30
Deleted: 30
Deleted: 30
Deleted: 30
Deleted: 31
Deleted: 32
Deleted: 32
Deleted: 32
Deleted: 32
Deleted: 33
Deleted: 34
Deleted: 34

Deleted: 38

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" [2]—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, 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.

Comment: ITF: see comments above on "Platform".
This intro material needs to be revised to better summarize the goals of the OGSA WG.

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.

Comment: Perhaps this paragraph is where we should provide a more concrete definition of a profile.

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

Comment: ITF: Do we want to introduce this concept, or not? Again, I can't recall the outcome of our discussions on this topic.

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 §**Error! Reference source not found.**, we review briefly the scope of the OGSA-WG and the purpose of this document. In §2, we provide an overview of the use cases that we used to motivate requirements. In §3, we motivate and describe the OGSA Platform. In §**Error! Reference source not found.**, we provide more detailed descriptions of OGSA services.

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.

2   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. These use cases are collected in a companion document [REF]. The use cases have 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), but have provided useful input to the definition process.

2.1   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. *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.

  > **Comment:** This and the following two from Andreas and Hiro

- *Load Balancing* In many applications it is necessary to make sure make sure deadlines are met, or resources are used uniformly. These are both forms of load balancing that must be made possible by the underlying infrastructure.

- *Pricing*

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.

3   The OGSA Platform Service Taxonomy

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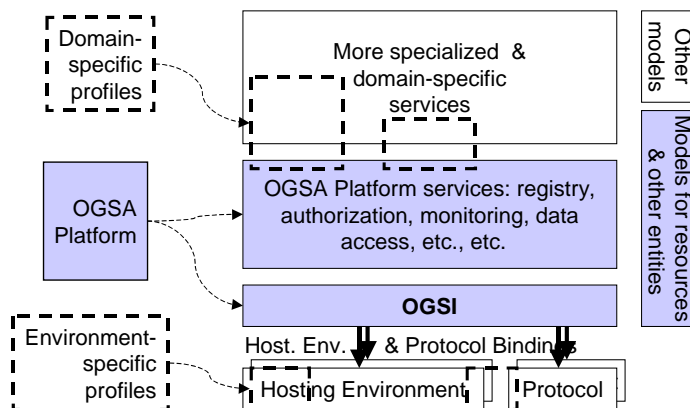
Figure 1: OGSA Platform components (shaded) and related profiles (dashed lines)

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 Environment Specific Profiles addressing issues such as the following. We mention these here for completeness; they are not discussed further in this document.

- *Protocol bindings*. Environment 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*. Environment 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.

The OGSA Platform defines functions that occur within a wide variety of Grid systems. We divide these functions into three broad groups: core, data- and information-related, and computation management-related.

3.1 *Core Services*

"Core" services are implementations of those functions which are generally used by a wide variety of higher-level services and that implement capabilities that are broadly useful. While dependencies between core services and non core services (by higher level functions) are likely, there is presently no known requirement that any particular core service be present in order to implement a Grid. The organization of these "core" services into the areas below is for the

convenience of explanation and is not meant to imply dependencies among functions that are grouped together.

3.1.1 Service Interaction

These are services primarily intended to provide interaction mechanism for the collection of services in the grid. They provide means for services to be registered and locate each other as well a mechanisms for composing multiple lower-level services into aggregations. They also include functions for deploying the software images that implement services in hosting environments and for collecting data about their operation for management, accounting, and billing purposes.

This service category includes the following services:

- Virtual organizations
- Service group and discovery services
- Service domain, service composition, orchestration, workflow
- Transaction

3.1.2 Service Management

These are functions for managing the services deployed in the distributed grid. Service Management automates and assists with a variety of installation, maintenance, monitoring, and troubleshooting tasks within a Grid system. This includes functions for provisioning and deploying the system components. It also include functions for collecting and exchanging data about the operations of the grid. This data is used for both "online" and "offline" management operations and includes information about faults, events, problem determination, auditing, metering, accounting and billing. Service management may also depend on models and schema that describe dependency relationships between different components, installation and provisioning steps and processes, and external capabilities.

This service category includes the following services:

- Metering and Accounting
- Installation, Deployment, and Provisoning
- Fault Management
- Problem Determination

3.1.3 Service Communication

The services provide the basic methods for services to communicate. They support several communication models that may be composed to permit effective inter-service communication.

This service category includes the following services:

- Distributed Logging
- Messaging and Queueing
- Event

3.1.4   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 GGF's OGSA-Sec working group, seeks to address these goals in a manner consistent with the security model that is currently being defined for the Web services framework 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.

As discussed in §3, 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 §**Error! Reference source not found.** 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.

3.1.4.1   List of detailed services

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.

Deleted: the Global Grid Forum's

Comment: Japanese

Deleted: !

Comment: Really usually referred to a Quality of Protection (QoP) in security paralance.
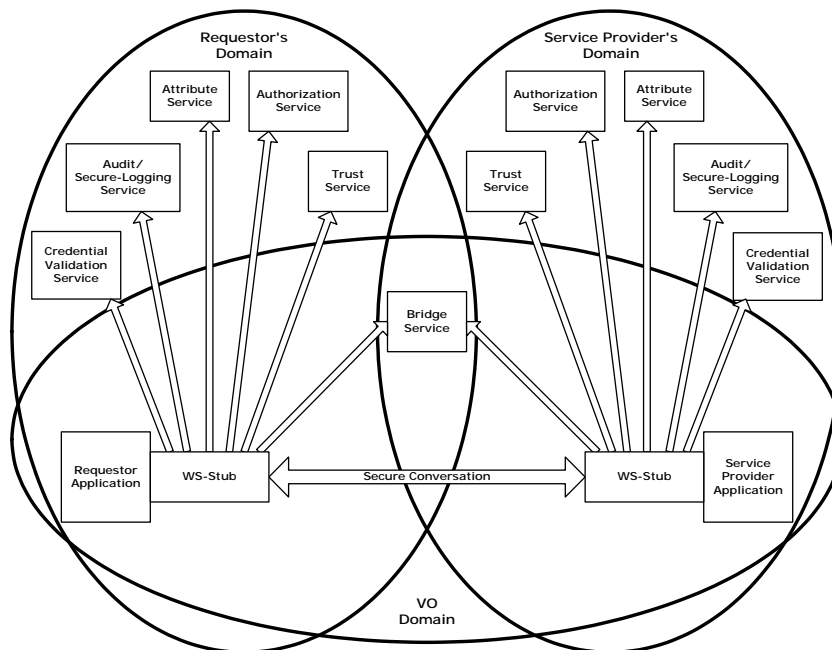
Figure 2: Security services in a virtual organization setting

The relationship between a requestor, service provider and many of the security services is depicted in Figure 2. 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.

3.2 *Data 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.

3.3 *Program Execution*

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

Figure 3 shows the Grid services required for Program Execution. These services include:

- Job Service – creates, monitors and controls compute jobs

- Reservation Service –guarantees resources are available for running a job

- Queuing Service – provides a service where administrators can customize and define scheduling policies at the VO level, and/or at the different resource manager levels

- Global Information Service – allows for the propagation of information between resource managers and the metascheduler.

- Resource Manager Adapter Service (RM Adapter) – provides a Grid service interface that bridges the Grid service protocol and resource managers (e.g. LSF or PBS).

Figure 3 Program Execution Architecture and Services

3.4   Resource Management

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

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

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

Reservation and Scheduling Services. From Hiro and Andreas: Reservation Services provide the mechanism to make resources reservation at a particular time duration. Scheduling Services provide the mechanism to scheduling tasks according to their priorities.

Deployment Services. From Hiro and Andreas: Deploy necessary software (OS, middleware, application) and data into hosting environment

ITF: OVERALL COMMENTS ON SECTION 3:

- There isn't enough detail provided here to know whether the specified "services" (interfaces?) are sufficient or not (for what?)

Deleted: ¶

- What about "common models" and other aspects of an overall solution, like transport protocols?

- (Broader comment): what about Web services and our relationship to that.

4   Service Relationships

ITF: Use "Body Text" as basic paragraph font.

The previous Chapter categorized services in the OGSA platform as follows:
- It showed the services that:
  - Provide a complete set of interfaces that implements a functionality related to (or needed by) Grid computing
  - Provide high-level functionality
  - Provide a user's (end-user, management software, resource managers, etc) view of the OGSA platform
- The services were classified according to a taxonomy, in which two services are related (i.e., put on the same category) if their purpose and functionality are similar. However, this taxonomy does not show the relationships that exist between the services when these services are used in practice.

This chapter gives a different categorization of the OGSA platform as follows:
- It provides a provider view of the services (instead of the user's view)
- It defines types of relationships between services, and organizes the services according to these relationships
- It introduces a new class of services, the *foundation services*

4.1   Types of Relationships

The services in the OGSA platform have two kinds of relationships between themselves:
- "Uses" relationship: a first service accesses the interface of a second service to use the functionality provided by this second service. For instance, many services will use the handle resolver service to convert GSHs to GSRs.
- "Extends" relationship: a first service extends the functionality provided by a second service by using portType extensibility. A simple example of this relationship is an event service that extends the OGSI notification functionality. Another example is a registry service that extends the service group functionality of OGSI.

4.2   Foundation Services

The foundation services have the following characteristics:
- They provide functionalities that are basic and focused
- They provide functionalities which are common to (and used by) several high-level services
- Their main intent is to be used through the "extends" relationship

The functionality provided by a given foundation service is present (through extension) in several high-level services. As a consequence, the functionalities of the foundation services permeate the high-level services, being pervasive on the OGSA platform. For this reason, they don't fit into (are not shown as part of) the taxonomy of the previous chapter.

Deleted: <#>¶
¶
<#>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 th [... [3]]

Deleted: ¶

Comment: Define what this means

Comment: I'm not necessarily opposed to these, but it is a fundamental new concept and so we need to make sure it is well motivated. And, if it is, then we should introduce it earlier.

Deleted: ¶

Comment: These are not well-defined terms!

Due to their nature, the foundation services form the lower layer of the service relationships. However, a foundation service may use or extend other foundation services (i.e., there is more than one layer of foundation services).

As the high-level services are organized and categorized, and their functionality is defined in more detail, common functionalities among these services will increasingly appear. These functionalities should be re-defined as foundation services in order to simplify the OGSA platform. As a consequence, as the work on the definition of the OGSA platform progresses, the number of foundation services should increase.

Currently, the foundation services of the OGSA platform are listed below (full descriptions are given in Chapter 5):
- OGSI: defines Grid services and the basic mechanisms for creating, managing, and exchanging information between them.
- WS-Agreement: provides a set of interfaces that support the negotiation of policies, service level agreements, reservations, etc., and maps the related agreements to Grid services.
- Common Management Model (CMM): provides the manageability infrastructure for resources in the OGSA platform. CMM defines the base behavioral model for all resources and resource managers in the Grid, plus management functionality like relationships and lifecycle management.
- OGSA Data Services (or part of it): provides the basic functionality to manage data in a Grid environment.

Comment: Is this reference going to become a reference to CRM/WSMF?

The following are examples of areas in which we suspect that new fundamental services remain to be defined:
- Security
- Policy (the management of policies themselves, not the management through policies)
- Information services

Deleted: are expected to be created in

Deleted: the future

4.3 Relationships

[Figure]

[TBD, during or after face-to-face]

5   OGSA Platform Services

5.1   Handle 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.

5.1.1   List of detailed services
TBD.

5.1.2   Underlying services
TBD.

5.1.3   Related standards
TBD.

5.1.4   Owner WG
TBD.

5.2   Virtual Organizations

"Virtual Organizations" are a concept that supplies a "context" for operation of the grid that can be used to associate users, their requests, and resources. VO contexts permit the providers of grid resources to associate appropriate policy and agreements with the resources they contribute to the grid. Users that are then associated with (member of) the VO are able to exploit those resources consistent with those policies and agreements. Virtual Organization Management provides functions for the creation and management of VOs. These functions include mechanisms for associating users/groups with the VO, manipulation of user roles (administration, configuration, use etc.) within the VO, association of services (encapsulated resources) with the VO, attachment of agreements and polices to the VO as a whole or individual services within the VO. Finally creation of a VO requires a mechanism by which the "VO context" is referenced and associated with user requests. (This is most likely via a GSH since a "service" is the likely embodiment of the VO).

5.2.1   List of detailed services
TBD.

5.2.2   Underlying services
TBD.

5.2.3   Related standards
TBD.

5.2.4   Owner WG
TBD.

5.3   Service Groups and Discovery Services

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

| Deleted: ¶ |

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.

5.3.1   List of Detailed Services

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.

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.

Deleted: d

Deleted: s

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.

5.3.2 Underlying Services

TBD.

Deleted: s

5.3.3 Related Standards

TBD.

Deleted: s

5.3.4 Owner WG

TBD.

5.4 Service Domain, Service Composition, Orchestration, Workflow

<Service Domain> 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.

<Service orchestration or workflow> 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.

<Service Composition>

5.4.1   List of detailed services

<Service Domain> 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.

<Service orchestration, workflow, Service Composition >

5.4.2   Underlying services

TBD.

5.4.3   Related standard**s**

TBD.

5.4.4   <Owner WG>

TBD.

5.5   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.

5.5.1   List of detailed services

5.5.2   Underlying services

5.5.3   Related standards

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.

5.5.4   <Owner WG>

TBD

5.6   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.

5.6.1   List of detailed services

- *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.

- *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.

- *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.

- *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.

5.6.2   Underlying services

5.6.3   Related standard

5.6.4   Owner-WG

Current RUS-WG covers little bit lower part of these services.

5.7   Installation, Deployment, and Provisioning

Computer CPUs, applications, licenses, storage, networks, and instruments are all Grid resources that require installation, deployment and 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.

5.7.1   List of detailed services

5.7.2   Underlying services

5.7.3   Related standards

5.7.4   Owner WG
TBD.

5.8   Fault Management

5.8.1   List of detailed services

5.8.2   Underlying services

5.8.3   Related standards

5.8.4   Owner WG
TBD.

5.9   Problem Determination

5.9.1   List of detailed services

5.9.2   Underlying services

5.9.3   Related standards

5.9.4   Owner WG
TBD.

5.10 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*.

5.10.1 List of detailed 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.
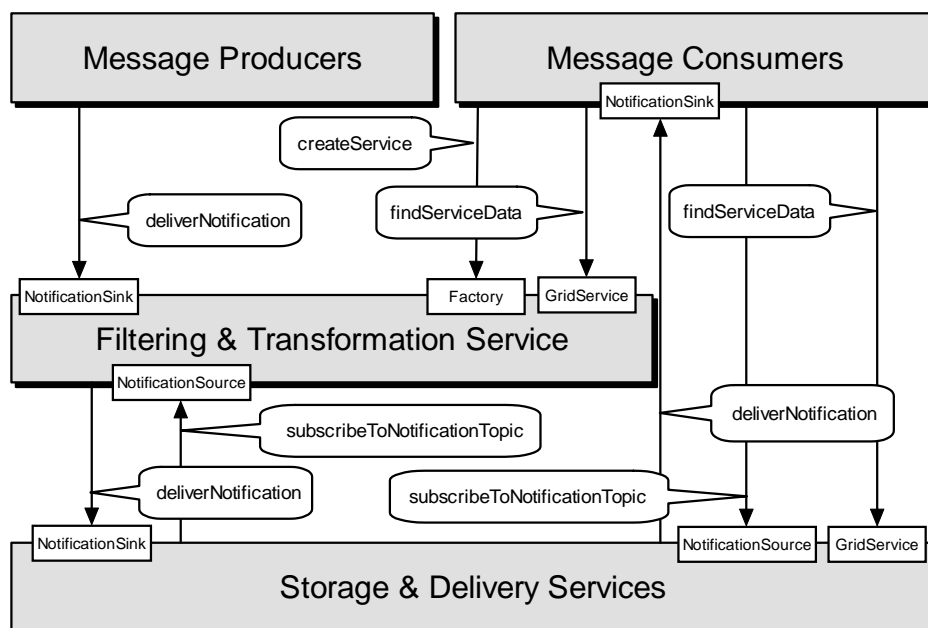
Figure 4: Schematic of a messaging service architecture

5.10.2 Underlying services

5.10.3 Related standards

5.10.4 Owner WG

5.11 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.

5.11.1 List of detailed services

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.

5.11.2 Underlying services

5.11.3 Related standard

5.11.4 <Owner WG>

5.12 Event

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?

5.12.1 List of detailed services

- 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..

5.12.2 Underlying services

5.12.3 Related standard

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?

5.12.4 <Owner WG>

No

5.13 Policy and Agreements

These services create a general framework for creation, administration and management of policies and agreements for system operation, security, resource allocation, etc. as well as an infrastructure for "policy aware" services to use the set of defined and managed policies to govern their operation. These services do not actually enforce policies but permit policies to be managed and delivered to resource managers that can interpret and operate on them. Agreements (OGSI-Agreement Specification) provide a mechanism for the representation and negotiation of ***terms*** between service providers and their clients (either user requests or other services). These terms include specifications of functional, performance, and quality requirements/objectives that the suppliers and consumers exchange and that they can then use to influence their interactions. The Agreement mechanism provides a general expression framework for these terms but leaves the specification of particular terms to individual disciplines such as performance, security, data quality etc. Agreements also contain information on priority, costs, penalties or consequences associated with violation of the "contract" a negotiated agreement represents as well as information on how the service provider and consumer have decided to measure compliance with the agreement.

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)

5.13.1 List of detailed services

These interfaces provides a framework for creating, managing, validating, distributing, transforming, resolving, and enforcing policies within a distributed environment.

- 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: §1)

The Policy Service Manager controls access to the policy repository. It also controls when notifications of policy changes are sent out so that multiple updates can be made and notifications are only sent after all updates are complete. The Policy Service Agent is a management service that other "policy aware" services depend on for delivery of thier 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 configuration changes in the resource(s) they manage, by using the Common Resource Model mechanisms referred to in §1. 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.
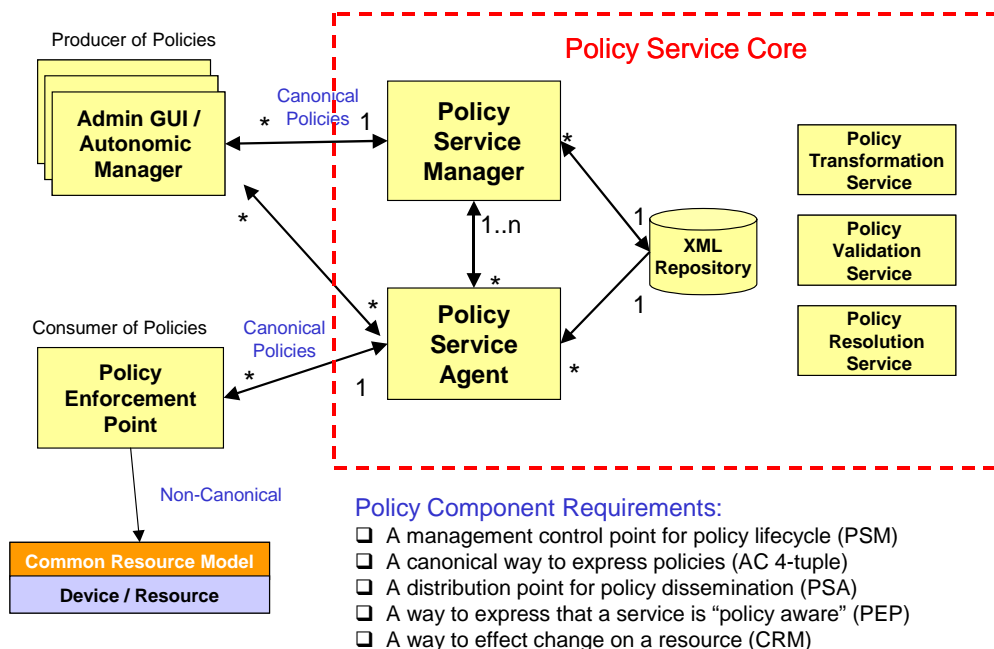
Figure 5: A set of potential policy service components

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.

5.13.2 Underlying services

5.13.3 Related standards

WS-Policy, etc.

5.13.4 Owner WG

~~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.~~

5.14 Base Data Interfaces

ITF: This material is taken from the "OGSA Data Services" position paper that was presented at the recent OGSA-DAI F2F. Discussion at that meeting will motivate revisions to that position paper, and to the text below.]

OGSA data interfaces are intended to enable a service-oriented treatment of data, so that data can be be treated in the same way as other resources within the Web/Grid services architecture. Thus, for example, we can integrate data into registries and coordinate operations on data using service orchestration mechanisms. A service-oriented treatment of data also allows us to use Open Grid Services Infrastructure (OGSI) Grid Service Handles as global names for data, manage the lifetime of dynamically created data by using OGSI lifetime management mechanisms, and represent agreements concerning data access via WS-Agreement.

OGSA data services are intended to allow for the definition, application, and management of diverse abstractions—what we term *data virtualizations*—of underlying data sources. A data virtualization is represented by, and encapsulated in, a *data service*, an OGSI Grid service with SDEs that describe key parameters of the virtualization, and with *operations* that allow clients to inspect those SDEs, access the data using appropriate operations, derive new data virtualizations from old, and/or manage the data virtualization. For example, a file containing geographical data might be made accessible as an image via a data service that implements a "JPEG Image" virtualization, with SDEs defining size, resolution, and color characteristics, and operations provided for reading and modifying regions of the image. Another virtualization of the same data could present it as a relational database of coordinate-based information, with various specifics of the schema (e.g., table names, column names, types) as SDEs, and SQL as its operations for querying and updating the geographical data. In both cases, the data service implementation is responsible for managing the mapping to the underlying data source.

5.14.1 List of Detailed Interfaces

We define four *base data interfaces* (WSDL portTypes) that can be used to implement a variety of different data service behaviors:

- *DataDescription* defines OGSI service data elements representing key parameters of the data virtualization encapsulated by the data service.

- *DataAccess* provides operations to access and/or modify the contents of the data virtualization encapsulated by the data service.

- *DataFactory* provides an operation to create a new data service with a data virtualization derived from the data virtualization of the parent (factory) data service.

- *DataManagement* provides operations to monitor and manage the data service's data virtualization, including (depending on the implementation) the data sources (such as database management systems) that underlie the data service.

We specify that a data service is any OGSI-compliant Web service that implements one or more of these base data interfaces.

5.14.2 Underlying Interfaces

The definitions for these services build on and extend core OGSI interfaces, in particular GridService and Factory.

They are also expected to build on and extend OGSI-Agreement interfaces [1], which are used to incorporate agreements into the various data operations.

We also expect that (yet-to-be-defined) OGSA relationship management services will be used to represent and manage relationships among virtualizations, such as multiple virtualizations against the same data source, and dependencies between virtualizations.

5.14.3 Related Standards

Not sure about this.

5.14.4 Owner WG

OGSA-DAI.

5.15 Other Data Interfaces

A wide variety of higher-level data interfaces can and must be defined on top of the base data interfaces, to address functions such as:

- Data access and movement.
- Data replication and caching.
- Data and schema mediation.
- Metadata management and looking.

There are likely to be strong relationships to discovery, messaging, agreement, and coordination functions.

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.

5.15.1 Detailed List of Interfaces

The following is far from being a complete list.

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

Deleted: *Access Service*

Deleted: ¶

Deleted: *Data Replication*¶

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. 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.

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

5.15.2 Underlying Interfaces
Build on basic OGSI interfaces, in particular OGSI GridService and ServiceGroup.

5.15.3 Related Standards
TBD.

5.15.4 Owner WG
TBD.

5.16 Discovery Interfaces
Discovery interfaces address the need to be able to organize and search information about various sorts of entities in various ways. In an OGSA environment, it is normally recommended that entitities of whatever type be named by GSHs, thus discovery services are concerned with mapping from user-specified criteria to appropriate (GSHs). Different interface definitions and different implementation behaviors may differ according to how user requests are expressed, the information used to answer requests, and the mechanisms used to propagate and access that information.

Deleted: <#>**Data Caching Service**¶

Deleted: ¶
Metadeta catalog Service

Deleted: 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 Service*¶

5.16.1 Detailed List of Interfaces
Not clear.

5.16.2 Underlying Interfaces
Build on basic OGSI interfaces, in particular OGSI GridService and ServiceGroup.

5.16.3 Related Standards
HandleResolver?

5.16.4 Owner WG
TBD.

5.17 *Job Service*

The Job service provides an interface for placing jobs on a resource manager (i.e. representing a machine or a cluster), and for interacting with the job once it has been dispatched to the resource manager. The Job service provides basic matchmaking capabilities between the requirements of the job and the underlying resource manager available for running the job. More advanced Job services take into account more advanced job characteristics such as interactive execution, parallel jobs across resource managers, and jobs with requirements based on SLAs.

The interfaces provided by the Job service are:
- Manageability Interface ---
  - Supported Job Terms: defines a set of service data used to publish the job terms supported by this job service, including the job definition (command line and application name), resource requirements, execution environment, data staging, job control, scheduler directives, accounting and notification terms.
  - Workload Status: total number of jobs, statuses such as number of running, pending and suspended jobs.
- Job Submission – accept user job submission requests. The job can be submitted with a set of job terms in order to bind the job to a specific resource reservation, or submitted with the name of a specific resource manager on which to run the job. The job submission interface should support OGSI-Agreement protocol and should be compliant with the OGSI-agreement job terms. The initiator passes as parameters the set of job terms, including command line, resource requirements, execution environment, data staging, job control, scheduler directives, accounting and notification terms.
- Job Control – control the job after it has been instantiated. This would include the ability to suspend/resume, checkpoint, and kill the job.

The Job service makes use of information in the form of policies which are defined at the metaScheduler level, and resource information about available resource managers, queues, host, and job status as provided by the Global Information Service. The Job service uses the Resource Manager Adapter (RM Adapter) in order to submit jobs to the underlying resource manager, and in order to control the running jobs.

Deleted: *<#>Storage Service*
Storage. Storage can be modeled as a service just as any other resource. Grid storage interfaces can be represented as CRM services.

5.18 Reservation Service

allows end users or a Job service to reserve resources under the control of a resource manager to guarantee their availability to run a job. The service allows reservations on any type of resource (e.g. hosts, software licenses, or network bandwidth). Reservations can be specific (e.g. provide access to host "A" from noon to 5pm), or more general (e.g. provide access to 16 Linux cpus on Sunday). Once a reservation is made, a Job service can send a job to a resource manager that is attached to the provided reservation. Some of the policy decisions made by the Reservation service for use with a RM include notions of who can make reservations (e.g. admins only), how many hosts a particular user or user group can reserve at a time, when reservations can be made (i.e. "blackout periods"), and what types of hosts can be reserved.

The Reservation service provides the following interfaces:

- Manageability – define a set of service data that describes the details of a particular reservation, including resource terms, start time, end time, amount of the resource reserved and the authorized users

- Add Reservation – request a new reservation with a particular resource requirement, starting at a specific time for a given duration

- Delete Reservation – remove a reservation

The Reservation service makes use of information about the existing resource managers available, any policies that might be defined at the VO level, and will make use of a logging service to log reservations. It will use the RM Adapter interfaces to make reservations and to delete existing reservations.

5.19 Queuing Service

The Queuing service provides scheduling capability for jobs. Given a set of policies defined at the VO level, a Queuing service will map jobs to resource managers based on the defined policies. For example, a Queuing service might implement a fairshare policy that makes sure that all users within the VO get reasonable turnaround time on their jobs, rather than being starved out by other users' jobs ahead of them in the queue.

·   Manageability Interface - Define a set of service data for accessing QoS terms supported by the Queuing services. QoS terms for the Queuing service can include whether the service supports on-line or batch capabilities, average turn-around time for jobs, throughput guarantees, the ability to meet deadlines, and the ability to meet certain economic constraints.

5.19.1 Underlying services

- Global Information Service – allows for the propagation of information between resource managers and the metascheduler. It also provides facilities for other services to persist their state data.
- Resource Manager Adapter Service (RM Adapter) – provides a Grid service interface which bridges between the Grid service protocol and a given resource manager (e.g. LSF or PBS). The interface to a Resource Manager Adapter Service is a composition of the interfaces provided by the Job Service and Reservation Service, since both the Job Service and the Reservation Service use the services of the RM Adapter.

Deleted: ¶

Deleted: ¶

Comment: ITF: I pick on this subsection as an example of why I am not happy with this laundry list of services. It isn't clear to me that the text here provides any value. Sure, people are going to define queuing services of various sorts for different sorts of tasks. I don't think we know (or necessarily care?) what functions such services should provide. So does it really help to have a section on it?

(I'm happy to be persuaded otherwise.)

Deleted: ¶

- Persistent service: required for storing job, configuration and events.

5.19.2 Related standards

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

- OGSI-Agreement Job Terms and Reservation Terms

5.19.3 Owner WG

GRAAP: OGSI-Agreement

JSDL: Job Terms and Resource Reservation Terms

5.20 Open Grid Services Infrastructure

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.

> **Comment:** This should come earlier.

5.20.1 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.

5.20.2 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*.

5.21 WS-Agreement

[TBD]

5.22 Common Management Model

Comment: This has to be revised to reflect CRM/WSMF discussions.

A fundamental requirement of grid management infrastructure is the ability to define the resources and resource management functions of the system in a standard and interoperable way.

The capabilities of the grid management infrastructure rely on the ability to discover, compose, and interact with the resources and the resources managers responsible for them.

The Common Management Model (CMM) specification defines the base behavioral model for all resources and resource managers in the grid management infrastructure. A mechanism is defined by which resource managers can make use of detailed manageability information for a resource which may come from existing resource models and instrumentation, such as those expressed in CIM, JMX, SNMP, etc, combined with a set of canonical operations introduced by base CMM interfaces. The result is a manageable resource abstraction that introduces OGSI compliant operations over an exposed underlying base content model. CMM does not define yet another manageability (resource information) model.

The Common Management Model specification defines

- the base manageable resource interface – that which a resource or resource manager must provide to be manageable
- canonical lifecycle states, the transitions between the states, and the operations necessary for the transitions that complements OGSI lifetime service data
- the ability to represent relationships among manageable resources (instances and types) including a canonical set of relationship types
- lifecycle metadata (XML attributes) common to all types of managed resources for monitoring and control of service data and operations based on lifecycle state
- canonical services factored out from across multiple resources or domain specific resource managers, such as an operational port type (start/stop/pause/resume/quiesce)

Additional items that may become within the scope of the CMM specification (driven by its companion community practice document) are

- new data types or metadata to convey semantic meaning of manageability information, such as counter or gauge
- versioning information
- meta data to associate a metered usage (unit of measure) with manageability information
- classification of properties such as metric, configuration
- registries and locating fine-grained resources
- managed resource identifier

To summarize, CMM represents the proposed industry standard for representing manageable resources and resource managers within the grid management infrastructure. The CMM interfaces are OGSI compliant and are used as the base abstract interfaces from which specific manageable resource types are derived. Standardization of the base management behavior is required in order to integrate the vast number and types of resources and more limited set of resource managers introduced by multiple vendors.

5.22.1 List of interfaces

The interfaces is currently being defined.

5.22.2 Standard schema/document

The schema is currently being defined.

5.22.3 Underlying services

CMM extends OGSI.

5.22.4 Related services

The related services will become clearer as the CMM interfaces are defined.

5.22.5 Related standard

CMM is related to standard management models such as CIM, JMX, SNMP and also to Grid-related models such as the Unicore Resource Schema and GLUE.

CMM is also related to efforts in other standars bodies, such as in the OASIS WSDM TC and in the DMTF.

5.22.6 Owner WG(s)

CMM-WG.

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

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

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

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. Czajkowski, K., Dan, A., Rofrano, J., Tuecke, S. and Xu, M. Agreement-Based Grid Service Management (WS-Agreement). Global Grid Forum, Draft, 2003.
2. 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.
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.

2.1Use Cases

<< We agreed to delete this section since we have companion usecase document. >>

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.

Handle resolution service is included in Core service category.

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.

3.5 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.