

# The Open Grid Services Architecture

## *A Summary and Evaluation*

Peter Z. Kunszt  
IT Division - Database Group, CERN, 1211 Geneva Switzerland

April 16, 2002

*Commissioned by the UK e-Science Core Program*

## 1. Introduction and Summary

This write-up provides a short evaluation of the Open Grid Services Architecture (OGSA) and its Grid Service Specification as presented in [1] and [2]. We revisit the problem that is being addressed, the vision of OGSA and investigate its benefits and possible weaknesses in its current specification. We also discuss some of the issues OGSA is likely to face and conclude with a few more detailed comments.

OGSA is building upon an earlier work of the same authors, the Anatomy of the Grid [4], where they present an open Grid architecture and define the technologies and infrastructure of the Grid as “supporting the sharing and coordinated use of diverse resources in dynamic distributed virtual organizations (VOs)”. OGSA extends and complements the definitions given in this work by defining the architecture in terms of Grid Services and by aligning it with Web Service technologies.

Web Services are an emerging paradigm in distributed computing focusing on simple standards-based computing models. OGSA picks the Web Service Description Language (WSDL) [5], the Simple Object Access Protocol (SOAP) [6] and the Web Service Introspection Language (WSIL) [7] from the set of technologies offered by Web Services and capitalizes especially on WSDL. Since in the distributed world of Grid Computing the same problems arise as on the Internet concerning the description and discovery of services, OGSA certainly takes the right approach to embrace the extensible WSDL standard. OGSA defines a set of WSDL conventions and mechanisms and specifies the WSDL elements to be used in detail in [2].

OGSA defines and standardizes a set of (mostly) orthogonal multi-purpose communication primitives that can be combined and customized by specific clients and services to yield powerful behavior. OGSA defines standard interfaces (portTypes in WSDL terminology) for basic Grid Services. Each service has to implement the GridService interface to qualify as a Grid Service. However, in our opinion, requiring a single standard interface for all Grid components to ensure interoperability and systems integration is a choice that is questionable. Defining a Grid Service ‘base interface’ is problematic in terms of evolution of the specification. Having a tied-in interface makes it very hard to make changes later if all services have to be updated. In another OGSA

analysis document [3], D. Gannon et. al. already suggest changing this interface by splitting its functionality. Similar arguments are made in [9]. We may certainly anticipate more changes as OGSA evolves. Of course, a specific service like a registry service may mandate the implementation of certain interfaces to ensure interoperability.

We argue that the service introspection methods might also be discovered passively using the host environment or the registry (see also [9]). This choice of a mandatory interface also complicates integration with existing components since all those components need to implement this interface in order to qualify as a Grid Service.

OGSA is an implementation independent specification. Several implementations of this specification may be offered. For example, a future Globus distribution will offer one or more reference implementations.

As with all complex systems, the devil is in the details. OGSA touches on the points of scalability, manageability and interoperability but there are many questions that remain unanswered or are deferred to a later time. How are QoS metrics defined and described? How does a VO get bootstrapped? How are VO-wide and local security policies enforced? How to deal with requirements of high availability and consistency? How do local resources – hosting environments – control and monitor their services and guard themselves against malicious abuse? We will ask these and more questions in detail later in this document. It might well turn out – as it is pointed out in [2] – that OGSA might change significantly due to insights from one of these domains.

We feel that especially the discovery mechanisms need to be defined and architected in much more detail. As we are also very much interested in this exciting field, we plan to contribute to the OGSA discussion based on recent work [9] where a Web Service Discovery Architecture (WSDA) is defined aimed to address these issues.

To summarize: OGSA is definitely a long-needed step in the right direction to enable integration of Grid components. The main point of concern is that many essential components are either not well defined yet (like the service data) or missing (security mechanisms). The framework should be designed with the aim to have it modular, easily extensible and adaptable to change so that future mechanisms can be adopted as easily as the current Web Service paradigms. We understand that this is the first step in a long process to define an extensible Grid framework and OGSA is already a great achievement in having started off into the right direction. If the definition and evolution of OGSA is indeed an open community process, we are confident that the outstanding problems and issues can be addressed in future versions of the Open Grid Services Architecture. We hope that with this analysis and the questions we ask we can contribute to this process and help to move forward to solve the outstanding problems.

## **2. The Problem and the Vision**

In this section we summarize the problem that OGSA is trying to address and the vision of the Grid as it is presented in the Anatomy and Physiology of the Grid documents [4], [1]. We feel that this summary is necessary in order for us to understand to what extent the current OGSA service specification [2] addresses them and also to point to areas where still a lot needs to be done.

In the scientific domain there is a need to share resources to solve common problems among many individuals and institutions. This resource sharing should be:

FLEXIBLE,  
SECURE,  
COORDINATED,  
ROBUST,  
SCALABLE  
MEASURABLE (QoS metrics)  
TRANSPARENT to the users.

In commercial enterprise computing there is an increasing need for resource sharing as well, not just to solve common problems but also to enable enterprise-wide integration and business to business (B2B) partner collaboration. In addition to the qualities listed above distributed resources should be

INTEROPERABLE,  
MANAGEABLE,  
AVAILABLE and  
EXTENSIBLE.

Grid computing is trying to address this very long list of desirable properties. The definition of Virtual Organizations (VO) gives the first necessary semantics to address the problem systematically. Virtual Organizations are defined as a group of individuals and/or institutions that have a well-defined set of sharing and QoS rules associated with them. In the commercial context, VOs may be set up for anything between a single enterprise entity to complex B2B relationships. VOs may be defined in a very flexible way to address their highly specialized needs.

In the vision of the Grid end users in a VO can use the shared resources according to the rules defined by the VO and the resource providers. Different VOs may share the same resources. VOs can be dynamic, i.e. they may change in time both in scope and extension, and the Grid supports this change transparently to the users.

In the next section we investigate how OGSA addresses the challenge of supporting all the desirable properties and the resulting vision of Grids presented.

### **3. The OGSA Solution and its Extent**

The Open Grid Services Architecture views the Grid as an extensible set of Grid Services. A Grid Service is defined as "a Web Service that provides a set of well-defined interfaces and that follows specific conventions". Virtual Organizations are defined by their associated services, since the set of services 'instantiate' a VO. In OGSA each VO builds its infrastructure from existing Grid Service components and has the freedom to add custom components that have to implement the necessary interfaces to qualify as a Grid Service. We summarize the requirements that motivate the OGSA as stated in [1]

and [2] and the properties it is providing to the Grid (see previous section for the discussion of properties) in the table below:

	<i>Requirement</i>	<i>Property</i>
1	Support the creation, maintenance and application of ensembles of services maintained by VOs.	COORDINATED, TRANSPARENT, SCALABLE, EXTENSIBLE, <b>MANAGEABLE</b>
2	Support local and remote transparency with respect to invocation and location.	<b>TRANSPARENT</b> , FLEXIBLE
3	Support multiple protocol bindings (enabling e.g. localized optimization and protocol negotiation).	<b>FLEXIBLE</b> , TRANSPARENT, INTEROPERABLE, EXTENSIBLE
4	Support Virtualization in order to provide a common interface encapsulating the actual implementation.	<b>INTEROPERABLE</b> , TRANSPARENT , MANAGEABLE
5	Require mechanisms enabling interoperability.	<b>INTEROPERABLE</b>
6	Require standard semantics (like same conventions for error notification).	<b>INTEROPERABLE</b>
7	Support transience of services.	SCALABLE , MANAGEABLE , AVAILABLE , <b>FLEXIBLE</b>
8	Support upgradeability without disruption of the services.	MANAGEABLE , FLEXIBLE , <b>AVAILABLE</b>

Table 1: Requirements mentioned in [1],[2] for OGSA. We try to map each requirement to the properties Grid Systems are supposed to have from the previous section. The primary property is highlighted. An analysis of these requirements is given in the next section.

As mentioned before, OGSA focuses on the nature of services that makes up the Grid. In the OGSA existing Grid technologies are aligned with Web Service technologies in order to profit from existing capabilities of Web Services, including

- Service description and discovery
- Automatic generation of client and server code from service descriptions
- Binding of service descriptions to interoperable network protocols
- Compatibility with higher-level open standards, services and tools
- Broad industry support

OGSA thus relies upon existing Web Service technologies to address the requirements listed in Table 1. OGSA defines a preliminary set of Grid Service interfaces and capabilities that can be built upon. All of these interfaces are defined using WSDL to ensure a distributed service system based on standards. They are designed such that they address the following responsibilities:

<b><i>Responsibility</i></b>	<b><i>Interfaces</i></b>	<b><i>Address Requirement</i></b>
Information & Discovery	GridService.FindServiceData Registry.(Un)RegisterService HandleMap.FindByHandle <i>Service Data Elements</i> <i>Grid Service Handle and Grid Service Reference</i>	1,2,4,5,6
Dynamic service creation	Factory.CreateService	1,7
Lifetime Management	GridService.SetTerminationTime, GridService.Destroy	1,7
Notification	NotificationSource. (Un)SubscribeToNotificationTopic NotificationSink.DeliverNotification	5
Change Management	<i>CompatibilityAssertion data element</i>	8
Authentication	Impose on the Hosting Environment	SECURE
Reliable Invocation	Impose on Hosting Environment	ROBUST
Authorization and policy management	Defer to later	SECURE
Manageability	Defer to later	MANAGEABLE

Table 2: Current elements of OGSA and elements that are being mentioned by OGSA but are said to be dealt with elsewhere. We also refer to the requirement or to the Grid capability the elements address.

The GridService interface is imposed on all services. The other interfaces are optional, although there need to be many service instances present in order to be able to run a VO. Based on the services defined, higher-level services are envisaged such as data management services, workflow management, auditing, instrumentation and monitoring, problem determination and security protocol mapping services. This is a non-exhaustive list and is expected to grow in the future. There are some existing Globus

implementations that address some of the functionalities of higher-level services that will form part of the Globus Toolkit version 3.

OGSA introduces the concept of Grid Service Handles that are unique to a service and by which the service may be identified and looked up in the HandleMap. Grid Service Handles are immutable entities. The Grid Service Reference is then the description of the service instance in terms of (although not restricted to) WSDL and gives the user the possibility to refer to a mutable running instance of this service.

OGSA envisages to allow a hierarchical structure of service hosting environments as they demonstrate in their examples.

We analyze this first setting of the Open Grid Service Architecture below in detail.

## **4. Shortcomings and Open Issues**

We understand that the OGSA work has just begun and there is a long way to go before we have a complete architecture specification where all of the desired properties of Grids are addressed. This can only happen by having reference implementations and deployments of OGSA-compliant Grid middleware that will eventually expose the strengths and weaknesses of the architecture. OGSA will have to be refined and adjusted iteratively but this is a natural and healthy process and the first very important step has been taken. We feel that this step has been taken in the right direction and we hope that OGSA will be successful in its evolution into the open standards-based architecture that it set out to define.

In this section we point to shortcomings and open issues that open up potential areas of future development and analyze the current state of OGSA by addressing each of the Grid properties we listed in section 2. We identify the areas that we feel are not sufficiently addressed by OGSA at this stage by referring to the tables in the previous section. Please be aware that this is the state of OGSA as we understand it today and there may be developments we are not aware of.

### *Availability and Robustness*

These properties are only implicitly addressed in the OGSA description and are not too prominently dealt with (see tables in the previous section). We understand that the mechanisms of OGSA might greatly improve on the availability of services by introducing the Factory pattern but there needs to be further discussion of (higher-level?) services that deal with failing or unavailable instances and start up new ones automatically. By introducing Factories OGSA lays the ground for automated service startup and thus increases robustness and availability. We feel that it would greatly enhance this aspect of OGSA if some service data elements were defined in the Factory that would deal with failing instances and policies on how to restart them.

We need more discussion on how the Grid Services should behave if some kind of failure occurs. What happens if a service has been unavailable for a given time? How to deal with service overload? What if the network of a VO becomes partitioned? What happens if the Factory or the Registry is suddenly unavailable? This also touches a bit on the desired property of scalability.

## *Scalability*

By designing OGSA for explicitly supporting transient services and dynamic service creation and by using soft state updates, we think that this property is dealt with sufficiently at this stage. In Table 1, scalability is dealt with in requirement 1 and 7 which in turn are addressed by several interfaces (see Table 2). We will have to see in the reference implementations and deployments of OGSA where the bottlenecks are and whether any architectural issues arise because of an eventual lack in scalability. As we have mentioned before, the robustness of the system needs to be ensured also in a scalable manner.

## *Measurability*

In the introduction of [1] it is mentioned that each VO needs certain levels of QoS to be achieved and that they may be measured in many different ways. The ability to set up VOs fulfilling many different QoS requirements is highlighted as one of the most desirable properties of Grids. OGSA does not elaborate further on QoS metrics: The Grid property MEASURABLE is not addressed by any OGSA requirement in Table 1. We agree that this area is a fundamental one to the success of Grids in general and needs to be dealt with in the very near future.

There need to be not just agreed metrics of QoS but each Grid Service needs to define how it enhances or decreases certain QoS metrics. There might be the need to define a QoS namespace to be able to query this property of services more easily. Each Service also needs to declare its own internal QoS metrics and give a value in a specific instance if different instances of the same service can be set up such that the given metrics can change.

Measurability is also very important when a VO defines its own set of necessary services or when it analyzes its own state and optimizes its performance due to changes in its nature or requirements. In order to define, bootstrap and optimize a VO it is essential to have QoS metrics by which the VO can measure itself.

## *Integration*

OGSA starts out with this point on integration of services. There is a need to integrate services not just within but also across VOs. OGSA solves this problem by defining the GridService interface and requiring all services to implement it. But how this is achieved with OGSA in detail and especially across VO boundaries is not detailed. A lot of effort still needs to be put into the exact mechanisms and definition of common semantics that integration of services (across VOs) may be achieved. An example of common semantics is mentioned – a common error messaging strategy – but even this single example is not exploited further in the document, hence we feel that there is a lot of work still to be done.

The three examples of integration of services given in chapter 5 are very useful and give us an idea how integration may be achieved. We would like to see a much more detailed discussion how they actually might work in practice but we understand that this is best dealt with using the future reference implementations and deployments of OGSA.

## *Security*

This issue is touched but not elaborated on sufficiently. The hosting environment gets the burden of authentication – which is reasonable – but there is no discussion on how local and VO-wide security policies are enforced also on authentication. Is there the need for a Grid Service that deals with these issues or should each of the services have an interface addressing this, making it part of the GridService base interface? New developments in this area are necessary.

By relying on Web Services, the strong industrial drive to come to a solution in this area will help speed up the process to design a suitable security infrastructure. Recent press releases by Microsoft and IBM have indicated the industry's commitment in this area. There needs to be a lot of effort put into this domain also from the Grid community to check how existing Grid security infrastructures might interoperate with Web Service security mechanisms.

Security will have to be dealt with very soon within OGSA since it will depend on the success of the underlying security framework. Open questions include: How are VO-wide policies applied? How are local security policies enforced? What is the role of hosting environments? How is an audit performed? Can a user belong to more than one VO and use both resources even if the security mechanisms differ?

## *Interoperability and Compatibility*

Interoperability is explicitly mentioned as a requirement and it is one of the driving concepts behind OGSA. Web Services, including Grid Services, are designed such that the modules are highly interoperable. There is no uniform protocol required that each service has to speak. WSDL descriptions are there to ensure interoperability.

The notification framework for passing messages between Grid Services is addressing this property explicitly. It resembles the concepts of the Grid Monitoring Architecture [10] but without the possibility to register notification sources with the target (sink) of the notification event. Interoperability is very closely related to discovery because services that need to interoperate have to discover among other things which common protocols they can use and whether there are issues of compatibility.

OGSA addresses compatibility by having Service Description elements that declare compatibility to past versions. What is missing, as it has been pointed out in [3], is the capability to declare forward compatibility, not just backward (in)compatibility. But the necessary elements to discover compatibility are there.

## *Service Discovery*

As mentioned before, users of many services and services that want to interoperate need to get hold of the service descriptions to discover which services meet their needs or which services are still missing to achieve a given QoS. But how does the user know how to get hold of these descriptions? The answer of OGSA is the Registry and the HandleMap. The Registry needs to be searched to find the Grid Service Handles of the services that fulfill the user requirements – formulated in a query if necessary. The

HandleMap then can be contacted to retrieve the detailed description of the services in question.

By holding a Grid Service Handle one can get at the corresponding (WSDL) description and the HandleMap is bound to the HTTP(S) protocol to assure availability of the description without another necessary discovery step.

This whole mechanism however leads us back to the Service Discovery problem: How do we get the GSH of the relevant registries in the first place?

Also, it is not clear to us from the current documents how GSH to GSR mappings are performed outside the home HandleMap. How are remote HandleMaps kept in sync with the home HandleMap?

But more fundamentally, this touches again on unified QoS metrics and Service Data elements. How is it possible to get the handles of the registries that we can query to get a set of services that we might be interested in i.e. how do we find the registry or registries relevant for a given query? How is a query formulated to do so? OGSA briefly mentions the use of XQuery [8] to query the Service Data of the Registry. In a large Grid spanning administrative domains, service data is partitioned over one or more registries nodes, for reasons including autonomy, scalability, availability, performance and security. This needs to be clarified in the future.

Another interesting design decision of OGSA is that the Grid Service Handle is strongly coupled to the home HandleMap. This has several implications. The home HandleMap needs to point to a valid URL at all times. If the service is moved then all GSH will have to deal with the update latency in the DNS that might also impact on service discovery (mapping of GSH into GSR). Also, we do not understand why the GSH is a restricted URL and may not have extensions (i.e. point to a server page or have predefined HTTP GET options).

### *Manageability*

Manageability is also just touched upon as a desired element but is deferred to a later time. The idea of unified monitoring and controlling mechanisms is there but not further exploited. Higher-level services are supposed to deal with this, but how are they to interact and be integrated with each other? Having uniform monitoring and control mechanisms might impose stricter requirements on the architecture so this is worth giving more thought. The questions one might ask include: How can a VO be created and bootstrapped? How can it be changed? How can individual users monitor and control their Grid sessions?

### *Changeability*

OGSA sets out to address the Grid vision in a very specific way. This is necessary in order to make use of the existing Web Service technology but it also ties itself to a few conventions that are not necessarily in line with the vision of the Grid as presented in section 2. The strongest convention is to require that each Grid Service implements the GridService interface. This is a design choice that for a heterogeneous system like the grid might seem very valuable but it compromises the desired properties of FLEXIBILITY and EXTENSIBILITY. Also it makes it difficult for existing or legacy services to be

integrated in the Grid or be viewed as a Grid Service since first they have to implement the GridService interface. If the interface were to be changed due to major architectural restructurings in the future, all services would have to change as well, compromising the AVAILABILITY of the system. Already now there are suggestions to change this basic interface [3]. There are other means of dealing with unified service discovery methods, as described for example in [9].

### *Summary*

We summarize our evaluation of how OGSA deals with the different Grid properties in its current state in the table below. For details see the discussion in this section.

<i>Property</i>	<i>OGSA state</i>
FLEXIBLE	Well addressed
SECURE	Needs to be addressed soon
COORDINATED	Need more mechanisms
ROBUST	Needs mechanisms
SCALABLE	Well addressed
MEASURABLE	Needs fundamental metric definitions
TRANSPARENT	End user use cases are missing how this is achieved in detail
INTEROPERABLE	Defined, needs reality check
MANAGEABLE	Needs more work
AVAILABLE	Needs more work
EXTENSIBLE	Well defined

## **5. Acknowledgements**

I'd like to thank Wolfgang Hoschek for carefully reading a first version of this report, providing corrections and references and for many useful discussions. I'd also like to thank Heinz and Kurt Stockinger and Leanne Guy for discussions and insights.

## **6. References**

- [1] Ian Foster, Carl Kesselman, Jeffrey M. Nick and Steven Tuecke. The Physiology of the Grid. An Open Grid Services Architecture for Distributed Systems Integration. Feb.17 2002, <http://www.globus.org/ogsa/>

- [2] Steven Tuecke, Karl Czajkowski, Ian Foster, Jeffrey Frey, Steve Graham and Carl Kesselman. Grid Service Specification. Feb. 15 2002, <http://www.globus.org/ogsa>
- [3] Dennis Gannon et. al. An Analysis of the Open Grid Services Architecture. From the GGF-OGSI WG mailinglist.
- [4] Ian Foster, Carl Kesselman and Steven Tuecke. The Anatomy of the Grid, International Journal of Supercomputer Applications, 15(3), 2001.
- [5] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana. Web Services Description Language 1.1. W3C Note 15, 2001 <http://www.w3.org/TR/wsdl>
- [6] World Wide Web Consortium. Simple Object Access Protocol (SOAP) 1.1, W3C Note 8, 2000
- [7] P. Brittenham, An Overview of the Web Services Inspection Language, 2001, <http://www.ibm.com/developerworks/webservices/library/ws-wslover>
- [8] World Wide Web Consortium. XQuery 1.0: An XML Query Language, W3C Working Draft, December 2001.
- [9] Wolfgang Hoschek, A Unified Peer-to-Peer Database Framework for XQueries over Dynamic Distributed Content and its Application For Scalable Service Discovery, PhD Thesis, Technical University of Vienna, Austria, 2002. <http://edms.cern.ch/file/341826/1/phd2002-hoschek.pdf>
- [10] Brian Tierney, Ruth Aydt, Dan Gunter, Warren Smith, Valerie Taylor, Rich Wolski and Martin Swany. A Grid Monitoring Architecture, Grid Forum Working Draft GWD-Perf-16-2, January 2002.