



# Grid watch: Open standards architecture at the GGF

Level: Introductory

Thomas Myer ([tom@tripleddogdaremedia.com](mailto:tom@tripleddogdaremedia.com)), Principal, Triple Dog Dare Media

18 Nov 2003

This installment of "Grid watch" provides a quick overview of OGSA, OGSF, and other architecture-related initiatives at GGF.

In the first installment of this column, I gave you a brief overview of the Global Grid Forum (GGF). Now I'll turn my attention to grid architecture, a topic I find extremely hard to talk about. It's not that I think architecture is boring or unnecessary. Quite the contrary. It's just a huge, rambling, complex topic, and my job here is to pick out what's important to the developer community without getting too lost in the weeds.

## So what's so important about grid architecture?

Before we talk about grid architecture, I want to take a step back and ask, "What's in it for developers? Why is this important?"

Grid architecture is important for precisely the same reasons that real-world architecture is important. Like standing structures, grids can:

- Come in all sizes
- Consist of many different kinds of materials
- Be used for different kinds of purposes
- Have different levels of traffic (throughput) and capacity
- Require different levels of security
- Require different types and levels of infrastructure

Computational grids are very complex webs of computing resources. Systems in a grid can be running different operating systems on different hardware configurations, be owned by different organizations, contain different levels of available resources to share, and so forth.

A successful grid architecture can help grids manage resources across this heterogeneous environment. A good architecture should provide open, published interfaces that allow all the different grid components to interact. With any luck, a good architecture might even help deliver one of the holy grails of grid computing -- quality of service (QoS).

So that's what's in it for you: good architectures mean good grids.

## The OGSA

The Architecture Area ("Areas" are broad organizational structures within the GGF) hosts a number of working and research groups, each representing different architectural approaches to building grids. The group with the highest profile is working on the Open Grid Services Architecture (OGSA).

OGSA (pronounced 'awg-sah' by GGFers) has been designed to tackle all the things we discussed in our introduction with an added twist: much of it has been built on existing Web services standards (particularly SOAP). If you understand Web services, then you shouldn't have too much trouble making the leap to OGSA-based grid. (And if you don't yet understand Web services, take some time to examine the developerWorks [Web services zone](#)).

The best way to look at OGSA is to imagine a layered cake. Those of you familiar with the OSI stack will have no trouble with this image. At the bottom of the OGSA stack are physical resources. Resources can be servers, storage devices, and networks. Sitting right above the physical resources are logical resources, which are usually collections of middleware applications -- databases, file systems, workflow managers. Basically, these are any tools that help the grid virtualize and aggregate available physical resources.

Now here comes the Web services layer. To this layer, all resources -- whether logical or physical -- are merely services. Although it may seem a little strange to think of a server's available CPU cycles as a service instead of a thing, it makes perfect sense in the Web services world.

The Web services layer provides a solid base of support services. For example, it has built-in features for discovering what services are available, and ways to ask a service to describe itself so that it can be easily invoked. Furthermore, it has standard ways to transport a service request and a service response.

But just because Web services provide so many great tools doesn't mean they can be used off the shelf. Most have to be extended, but more about that in another section.

Sitting on top of the Web services layer is the grid services layer. This is where you'll find data extraction, program execution, monitoring, and other core services. This is an area that bears watching, as it is being defined at this very moment (and this process will continue for some time, I daresay). Another note to make at this point is the importance of the Open Grid Services Infrastructure (OGSI). It is quite literally one of the first attempts to bridge the Web services layer with the grid services layer in OGSA while at the same time extending core Web services.

As soon as the grid services layer starts firming up, you'll start to see individual applications appear in the topmost layer, known as the grid applications layer. This is where, as developers, we'll be able to build on the stack beneath us. You can draw an analogy between this stack of layers and the OSI stack -- it's pretty easy to build an e-mail application once someone has figured out the SMTP protocol.

---

## Like Web services, but not quite

O.K., so far I've been telling you that many concepts behind OGSA are built on top of Web services standards. In many ways, grids behave just like Web services; in other words, a system asks another system for something and gets back a response.

However, there are a couple of inherent characteristics of Web services that make them unsuitable for grid environments. First of all, Web services are stateless -- they don't remember from invocation to invocation what you've requested (or received). If you were doing a series of related calculations or queries (very likely in a grid environment), you'd have to figure out how to pass results of one calculation along to the next invocation.

Then there's the problem of transience. It so happens that all Web services outlive their clients. In other words, they are non-transient. Data available to one client is potentially viewable (or modifiable) by another client.

The OGSI takes these (and other) issues into account. Essentially, OGSI extends Web services, making transient services and state management in grid possible. Furthermore, one could argue that the way that

OGSI does what it does allows for the building of robust service-oriented architectures (or SOA), which is something that will make your CIO or IT director very happy.

For now, though, let's zoom in on OGSI particulars.

---

## Brief overview of OGSI

OGSI, like anything else about grid computing, is fairly analogous to other computing concepts. In any OGSI-compliant grid service, clients talk to factories that build service instances. Each service instance has a unique Grid Service Handle (GSH), and is therefore uniquely identifiable in the system. (A GSH is fairly analogous to a URI in the Web services world). If the client needs to communicate with the service, it resolves the GSH into a Grid Service Reference (GSR) that it uses to contact the service instance created by the factory. When a client wants to destroy the service instance, it can do so through an explicit destroy operation, or it can simply rely on the lifetime-based garbage collection built into OGSI (more on that a little later).

Because these instances are transient, they will eventually be destroyed. The lifetime of an instance can vary from application to application, and it can be extended or curtailed as needed through interfaces provided by OGSI. Generally speaking, an instance should live only as long as it is needed by a client. This way, every client has its own personal instance to work with. However, several clients can share a single instance, and it's nice to be able to self-destruct an instance after a period of inactivity through careful lifetime management.

Each grid service contains within it Service Data, a set of data that describes a grid service. The existence of Service Data allows introspection -- the ability of a grid service to describe itself to an outside entity.

Another important aspect of a grid service is a portType. A grid services portType is virtually the same as a Web services portType. It functions as an interface for messages passing back and forth between endpoints. The required core GridServices portType can be used to find and query a grid service, figure out expiration time for a service, and other basic functionality. The Factory portType specializes in creating new services. Specialized portTypes for security, lifecycle management, job execution, and other uses can extend the core portTypes.

Because portTypes and Service Data are both represented as XML (and specifically, as WSDL documents), OGSI-compliant grids inherit from Web services a very rich and expressive set of tools and techniques that support the exposure of legacy systems as Web services. The grid service extensions then allow these exposed services to be integrated and shared across multiple administrative domains (within or across corporate boundaries).

It's important to note that OGSI is still a "Proposed Recommendation." It can become a recommendation only when several interoperable implementations exist. Only then can OGSI be considered "done."

---

## So what's the difference between an architecture and infrastructure?

Those of you who are still with me are probably wondering what's the difference between OGSA and OGSI; or, more generally, what's the difference between an architecture and an infrastructure. An architecture like OGSA defines a grid service, and an infrastructure like OGSI can be implemented by individual tools to create an actual grid service.

For example, if you wanted a bridge built, you'd hire an architect. This architect would draw up all the

plans, but she would also work with a structural engineer, who would be in charge of making sure the right materials were used in the building of the bridge. Finally, all of these plans and specifications would be given to a team of construction workers, who would build the bridge according to the blueprints and specifications.

If the bridge is our grid service, then the architect is OGSA, the structural engineer is OGSi, and the construction workers are implementation tools like Globus Toolkit 3, Unicore/GS, OGSi:Lite, or OGSi.NET.

---

## What else is going on in Architecture?

OGSA and OGSi aren't the only initiatives afoot in the Architecture area of GGF. Other working and research groups include:

- **Semantic Grid:** A research group monitoring Semantic Web initiatives and figuring out best practices and case studies as they apply to grid computing.
  - **Grid Protocol Architecture:** A research group whose primary focus is providing a "conceptual framework for discussing the interrelationships, completeness, and minimality of the protocol approach to grid services."
  - **Service Management Frameworks:** A research group that is examining and developing technical and management issues in the emerging Service Oriented Grid community. These issues include (but are not limited to) the interoperable dynamic registration, discovery, binding, and departure of services within different frameworks; and the mechanisms and meta-data to support autonomous service composition.
  - **Common Management Model:** This working group focuses on putting together a set of common standards for managing IT assets.
  - **New Productivity Initiative:** This working group will define a particular lightweight, high-level architecture for distributed computing management.
- 

## Summary

I've covered a lot of territory just now, and it's a lot to absorb. If it seems that there is a lot going on, that it's changing a lot, or that it's somewhat confusing, then you'd be right on all counts.

## Resources

- Having a good handle on Web services is the only way you'll make the transition to grid. The developerWorks Web services zone is a great place to start. Go to <http://www.ibm.com/developerworks/webservices/>.
- Likewise, knowledge of XML will keep you from falling too far behind. Check out the developerWorks XML zone at <http://www.ibm.com/developerworks/xml/>.
- Read up on OGSA and OGSi by becoming a member of GridForge at <http://forge.ggf.org>.
- Take a look at all the projects going on the GGF Architecture area at <https://forge.gridforum.org/projects/arch>.

- If you want to browse all the GGF working groups, go to [http://www.ggf.org/L\\_WG/wg.htm](http://www.ggf.org/L_WG/wg.htm).
- Joshy Joseph's article on OGSI-based computing for developers is a good resource. See <http://www-106.ibm.com/developerworks/library/gr-ogsi/>.
- Ian Foster's paper on the "Anatomy of the Grid" at <http://www.globus.org/research/papers/anatomy.pdf> is pretty much required reading, as is his paper on the "Physiology of the Grid" at <http://www.globus.org/research/papers/physiology.pdf>.
- For more details about OGSA, see "A visual tour of OGSA" at <http://www-106.ibm.com/developerworks/grid/library/gr-visual/>.
- Download the latest Globus Toolkit at [globus.org](http://globus.org).
- Unicore/GS is David Snelling's OGSI project based on the Unicore architecture ([www.unicore.org](http://www.unicore.org)).

## About the author



Thomas Myer is the founding principal of Triple Dog Dare Media, an Austin-TX based firm that specializes in building Web services, XML, and database applications. He can be reached at [tom@tripleddaremedia.com](mailto:tom@tripleddaremedia.com).