Open Grid Service Infrastructure (OGSI)

Prepared by: Dr M. Mirabi
• The Global Grid Forum has started a number of architecture standardization efforts in order to provide the required improved software interoperability, security (confidentiality, integrity, and availability), resource definition, resource discovery, policy, and grid manageability.

• The Global Grid Forum is a forum of researchers and practitioners working on grid computing, and a number of working groups are producing technical specs, documenting user experiences, and writing implementation guidelines.

• The need for open standards that define these interactions and foster interoperability between components supplied from different sources has been the motivation for the Open Grid Service Architecture/Open Grid Services Infrastructure (OGSA/OGSI) milestone documentation published by the Forum.
• The OGSI documentation was published in 2002 and a draft version of the OGSA was published late in 2003.
• The Globus Toolkit™ is an open architecture, open standards, commercial grade tool for building computational grids;
• Globus Toolkit™ is a widely cited, solid reference implementation of the OGSA/OGSI standards.
• OGSA is a service-oriented architecture (SOA).
• OGSI defines mechanisms for creating, managing, and exchanging information among grid services.
• 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 capability.
• OGSI specifications define the standard interfaces and behaviors of a grid service, building on a Web services base.
• This approach provides a common and open standards-based mechanism to access various grid services using existing industry standards such as SOAP, XML, and WS-Security.
Introduction

- The architecture for grid computing is defined in the Open Grid Services Architecture that describes the overall structure and the services to be provided in grid environments.
- The OGSI is a formal specification of the concepts described by the OGSA; it specifies a set of service primitives that define a nucleus of behavior common to all Grid Services.
Introduction

Basic functional model for grid environment.
Introduction

OGSA Architected Services

OGSI—Open Grid Services Infrastructure

- Discovery
- Lifecycle
- State Mgt.
- Service Groups
- Factory
- Notification
- Handle Map

Web services

OGSA reliance on OGSI.
Introduction

• The running of an individual service (for example, an information query) is called a service instance.

• **Grid Services include:**
  – Discovery
  – Lifecycle
  – State Management
  – Service Groups
  – Factory
  – Notification
  – Handle Map

• A “layering” approach is used to the extent possible in the definition of a grid architecture because it is advantageous for higher-level functions to use common lower-level functions.
What is OGSA/OGSI? A Practical View

- The OGSA aims at addressing standardization (for interoperability) by defining the basic framework of a grid application structure.

- The OGSA standard define what grid services are, what they should be capable of, and what technologies they are based on.

- The OGSA does not go into specifics of the technicalities of the specification; instead, the aim is to help classify what is and is not a grid system.

- It is called an architecture because it is mainly about describing and building a well-defined set of interfaces from which systems can be built, based on open standards such as WSDL.
• **The objectives of OGSA are to:**
  – Manage resources across distributed heterogeneous platforms.
  – Support QoS-oriented Service Level Agreements (SLAs). The topology of grids is often complex; the interactions between/among grid resources are almost invariably dynamic. It is critical that the grid provide robust services such as authorization, access control, and delegation.
  – Provide a common base for autonomic management. A grid can contain a plethora of resources, along with an abundance of combinations of resource configurations, conceivable resource-to-resource interactions, and a litany of changing state and failure modes. Intelligent self-regulation and autonomic management of these resources is highly desirable.
  – Define open, published interfaces and protocols for the interoperability of diverse resources. OGSA is an open standard managed by a standards body.
  – Exploit industry standard integration technologies and leverage existing solutions where appropriate. The foundation of OGSA is rooted in Web services, for example, SOAP and WSDL, are a major part of this specification.
What is OGSA/OGSI? A Practical View

- The OGSA relies on the definition of grid services in WSDL, which defines the operations names, parameters, and their types for grid service access.
- Based on the OGSI specification, a grid service instance is a Web service that conforms to a set of conventions expressed by the WSDL as service interfaces, extensions, and behaviors.
- Because the OGSI standard is based on a number of existing standards (XML, Web services, WSDL), it is an open and standards-based solution.
- This implies that, in the future, grid services can be built that are compatible with the OGSI standard, even though they may be based on a variety of different languages and platforms.
- Specifically, the grid service interface is described by WSDL, which defines how to use the service.
- A new tag, gsdl, has been added to the WSDL document for grid service description.
- The UDDI registry and WSIL document are used to locate grid services. The transport protocol SOAP is used to connect data and applications for accessing grid services.
- All services adhere to specified grid service interfaces and behaviors.
- Grid service interfaces correspond to portTypes in WSDL used in current Web services solutions.
• The interfaces of grid services address discovery, dynamic service-instance creation, lifetime management, notification, and manageability; the conventions of grid services address naming and upgrading issues.

• The standard interface of a grid service includes multiple bindings and implementations ("implementations" include Java and procedural/object-oriented computer programming languages).

• Grid services, such as the ones just cited, can, therefore, be deployed on different hosting environments, even different operating systems.

• The OGSA also provides a grid security mechanism to ensure that all the communications between services are secure.

• The definition of standard service interfaces and the identification of the protocol(s) are addressed in current OGSA specifications.

• A grid service capability could be comprised of computational resources, storage resources, networks, programs, databases, and so on.

• A grid service implements one or more interfaces, where each interface defines a set of method operations that is invoked by constructing a method call through, method signature adaptation using SOAP.
• The key principle of OGSA is that all grid resources (both logical and physical) are modeled as services.
• There are two main logical components of OGSA:
  – The Web-services-plus-OGSI layer
  – The OGSA-architected services layer.
Four main layers comprise the OGSA architecture:

- **Grid Applications Layer**: This layer is the user-visible layer. It supports user applications. Eventually, a “rich” set of grid-architected services is expected to be developed.

- **OGSA-architected grid services layer**: Services in this layer include: Discovery, Lifecycle, State management, Service Groups, Factory, Notification, and Handle Map. These services are based on the Web services layer. The GGF was working at press time to define many of these architected grid services in areas such as program execution, data services, and core services.
Four main layers comprise the OGSA architecture:

- **Web Services layer**, plus the OGSI extensions that define grid services: The OGSI specification defines grid services and builds on standard Web services technology. OGSI exploits the mechanisms of Web services such as XML and WSDL to specify standard interfaces, behaviors, and interaction for all grid resources. OGSI extends the definition of Web services to provide capabilities for dynamic, stateful, and manageable Web services that are required to model the resources of the grid.

- **Physical and logical resources layer**: The concept of resources is central to OGSA and to grid computing in general. Resources comprise the capabilities of the grid. Physical resources include servers, storage, and network. Above the physical resources are logical resources. Logical resources provide additional function by virtualizing and aggregating the resources in the physical layer. General-purpose middleware such as file systems, database managers, directories, and workflow managers provide these abstract services on top of the physical grid.
The OGSI introduces an interaction model for grid services. The interaction model provides a uniform way for software developers to model and interact with grid services by providing interfaces for discovery, life cycle, state management, creation and destruction, event notification, and reference management.
Factory:

- A mechanism (interface) that provides a way to create new grid services.

- Factories may create temporary instances of limited function, such as a scheduler creating a service to represent the execution of a particular job; or they may create longer-lived services such as a local replica of a frequently used data set.

- Not all grid services are created dynamically; for example, some services might be created as the result of an instance of a physical resource in the grid, such as a processor, storage, or network device.
• **Life cycle:**
  – A mechanism architected to prevent grid services from consuming resources indefinitely without requiring a large-scale distributed “garbage collection” scavenger.
  – Every grid service has a termination time set by the service creator or factory.
  – Because grid services may be transient, grid service instances are created with a specified lifetime.
  – The lifetime of any particular service instance can be negotiated and extended, as required, by components that are dependent on or manage that service.
  – In turn, a client with appropriate authorization can use termination time information to check the availability of the service; the client can also request to extend the current lease time by sending a keep-alive message to the service with a new termination time.
  – If the service accepts this request, the lease time can be extended to the new termination time requested by the client.
  – This soft-state life cycle is controlled by appropriate security and policy decisions of the service, and the service has the authority to control this behavior (for example, a service can arbitrarily terminate a service or can extend its termination time even while the client holds a service reference).
OGSA/OGSI Service Elements and Layered Model

• **State management:**
  – Grid services can have “state.”
  – The OGSI specifies a framework for representing this state, called service data, and a mechanism for inspecting or modifying that state, named Find/SetServiceData.
  – Furthermore, the OGSI requires a minimal amount of state in service data elements that every grid service must support, and requires that all services implement the Find/SetServiceData portType.

• **Service groups:**
  – Service groups are collections of grid services that are indexed (using service data described above) for some specific purpose. For example, they might be used to collect all the services that represent the resources in a particular cluster node within the grid.
OGSA/OGSI Service Elements and Layered Model

- **Notification:**
  - Services interact with one another by exchanging messages based on service invocation.
  - The state information (the service data described above) that is modeled for grid services changes as the system runs.
  - Many interactions between grid services require dynamic monitoring of changing state.
  - Notification applies a traditional publish/subscribe paradigm to this monitoring.
  - Grid services support an interface (NotificationSource) to permit other grid services (NotificationSink) to subscribe to changes.
  - The internal state of a grid service can keep track that this grid service has received one or zero messages.
  - This reliable message delivery mechanism guaranteed by the internal state can build business-oriented transactions.
Handle Map:

- This deals with service identity.
- When Factories are used to create a new instance of a Grid Service, the Factory returns the identity of the newly instantiated service.
- This identity is composed of two parts:
  - A Grid Service Handle (GSH)
  - A Grid Service Reference (GSR).
- A GSH provides a reference the grid service indefinitely;
- GSR can change within the grid services lifetime.
- The Handle Map interface provides a way to obtain a GSR given a GSH.
- The user application invokes create Grid Service requests on the Factory interface to create a new service instance.
- The newly created service instance associated with the grid service interface will be automatically allocated computing resources.
- Meanwhile, an initial lifetime of the instance can be specified before the service instance is created.
- The newly created service instance will keep the user credentials for performing further interactions with other systems over the Internet.
- The newly created grid service instance will be automatically assigned a globally unique name called the GSH, which is used to distinguish this specific service instance from other grid service instances.
<table>
<thead>
<tr>
<th>Port type</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GridService</td>
<td>FindServiceData</td>
<td>Query a variety of information about the grid service instance, including basic introspection information (handle, reference, primary key, home handle map: terms to be defined), richer per-interface information, and service-specific information (e.g., service instances known to a registry). Extensible support for various query languages.</td>
</tr>
<tr>
<td></td>
<td>SetTermination Time</td>
<td>Set (and get) termination time for grid service instance.</td>
</tr>
<tr>
<td></td>
<td>Destroy</td>
<td>Terminate grid service instance.</td>
</tr>
<tr>
<td>Notification-Source</td>
<td>SubscribeTo-NotificationTopic</td>
<td>Subscribe to notifications of service-related events, based on message type and interest statement. Allows for delivery via third-party messaging services.</td>
</tr>
<tr>
<td>Notification-Sink</td>
<td>Deliver Notification</td>
<td>Carry out asynchronous delivery of notification messages.</td>
</tr>
<tr>
<td>Registry</td>
<td>RegisterService</td>
<td>Conduct soft-state registration of grid service handles.</td>
</tr>
<tr>
<td></td>
<td>UnregisterService</td>
<td>Deregister a grid service handle.</td>
</tr>
<tr>
<td>Factory</td>
<td>CreateService</td>
<td>Create new grid service instance.</td>
</tr>
<tr>
<td>Handle Map</td>
<td>FindByHandle</td>
<td>Return grid service reference currently associated with supplied grid service handle.</td>
</tr>
</tbody>
</table>

*Interfaces for authorization, policy management, manageability, and likely other purposes remain to be defined.
OGSA/OGSI Service Elements and Layered Model

Drilling down an additional level of detail, one can further categorize grid-architected services into four categories:

- Grid core services
- Grid program execution services
- Grid data services
- Domain-specific services
OGSA/OGSI Service Elements and Layered Model

Grid Core Services

Service Management
- Registries and Discovery Services
  - Attribute Propagation and Query
  - Service Domain
- Service Orchestration
- Metering & Accounting
  - Installation & Deployment

Service Communication
- Managing and Queuing Services
- Event Services
- Distributed Secure Logging Service

Security
- Authentication
- Authorization & Access Control
- Credential Validation & Transformation

Policy Management
- Policy Service Manager
- Policy Agent
- Policy Transformation Service
- Policy Resolution Service
- Policy Validation Service
- Policy Administration Services and Negotiation Framework

Grid core services.
Grid core services are composed of four main types of services:
- Service management
- Service communication
- Policy management
- Security

Unlike the OGSI functions that are largely implemented as extensions to basic Web Services protocols and an interaction model, these core services are actually implemented as grid services (upon the OGSI base).

These services are considered core primarily because it is expected that they will be broadly exploited by most higher level services implemented either in support of program execution or data access, or as domain-specific services.
• **Service management:**

  – Service management provides functions that manage the services deployed in the distributed grid.
  
  – Service management automates a variety of installation, maintenance, monitoring, and troubleshooting tasks within a grid system.
  
  – Service management includes functions for provisioning and deploying the system components; it also includes functions for collecting and exchanging data about the operation 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 communication:

- This includes functions that support the basic methods for grid services to communicate with each other.
- These functions support several communication models that may be composed to enable effective inter service communication, including queued messages, publish–subscribe event notification, and reliable distributed logging.
- Grid services can be published to a UDDI registry, or WSIL documents;
- the UDDI registry becomes a central place to store such information about and locations for grid services that enables publishing and searching of trading partners’ businesses and their grid services.
- There are two types of UDDI registries: private and public.
- Application developers and/or service providers can publish the grid services to the public UDDI registries operated by IBM, Microsoft, HP, or SAP.
- If one wants to publish one’s own private or confidential grid services, one can use a private UDDI registry.
- As an alternative, for testing purposes or for small-scale integration, a developer can publish the company’s grid services to WSIL documents, since WSIL enables grid services discovery, deployment, and invocation without the need for a UDDI registry.
- WSIL provides the means for aggregating references of preexisting service description documents that have been authored in any number of formats;
- These inspection documents are then made available on a Web site.
• Policy services:

  – These create a general framework for creation, administration, and management of policies and agreements for system operation.

  – Policy services include policies governing security, resource allocation, and performance, as well as an infrastructure for “policy-aware” services to use policies to govern their operation.

  – Policy and agreement documents provide a mechanism for the representation and negotiation of terms between service providers and their clients (either user requests or other services); terms include specifications, requirements, and objectives for function, performance, and quality that the suppliers and consumers exchange and that they can then use to influence their interactions.
• **Security services:**

  - **Security services** support, integrate, and unify popular security models, mechanisms, protocols, and technologies in a way that enables a variety of systems to interoperate securely.

  - **These security services** enable and extend core Web services security protocols and bindings and provide service-oriented mechanisms for authentication, authorization, trust policy enforcement, credential transformation, and so on.
Grid Program Execution Services:
- Mechanisms for job scheduling and workload management implemented as part of this class of services are central to grid computing and the ability to virtualize processing resources.
- Although OGSI and core grid services are generally applicable to any distributed computing system, the grid program execution service class is unique to the grid model of distributed task execution that supports high-performance computing, parallelism, and distributed collaboration.
• Grid Data Services:
  – These interfaces support the concept of data virtualization and provide mechanisms related to distributed access to information of many types including databases, files, documents, content stores, and application-generated streams.
  – Services that comprise the grid data services class complement the computing virtualization conventions specified by program execution services (OGSA placing data resources on an equivalent level with computing resources).
  – Grid data services will exploit and virtualize data using placement methods like data replication, caching, and high-performance data movement to give applications required QoS access across the distributed grid.
  – Methods for federating multiple disparate, distributed data sources may also provide integration of data stored under differing schemas such as files and relational databases.
• Domain-Specific Services:
  
  – Domain-specific services will make use of the functionality that these services supply.
  
  – It is critical that the GGF working groups are concentrating on specifying a broad set of useful grid services that software vendors and developers can then begin to implement.
In the following figure, one notices that grid core services are likely to see a mix of open source reference implementations and vendor-provided “value added” implementations.

The bulk of technologies in this area will likely be commoditized, but areas like policy and security could provide vendors a chance to differentiate themselves.
Implementations of OGSI:

- For OGSA to grow in acceptance it needs to be implemented on multiple hosting platforms.
- The Globus Toolkit 3 (GT3) historically was the first full-scale implementation of the OGSI standard.
- GT3 was developed by the Globus Project, a research and development project focused on enabling the application of grid concepts to scientific, engineering, and commercial computing.
- It is expected that many of the OGSI implementations will be delivered via the open source development model and that existing reference implementations (GT3) will be used unmodified in appropriate hosting environments.
- GT3 is written in Java language using the J2EE framework; however, nothing limits OGSI from being implemented in other programming languages and hosted in other environments.
The following figure shows that a Java implementation of OGSI can be hosted on any of several J2EE environments (such as JBOSS, WebSphere, or BEA Weblogic). However, alternative platforms such as a traditional C or C++ environment or C# and Microsoft .NET are other possible hosting environments.
OGSI defines a component model that extends WSDL and XML schema definition to incorporate the concepts of:

- Stateful Web services
- Extension of Web services interfaces
- Asynchronous notification of state change
- References to instances of services
- Collections of service instances
- Service state data that augment the constraint capabilities of XML schema definition.
The OGSI specification (V1.0 at press time) defines the minimal, integrated set of extensions and interfaces necessary to support definition of the services that will compose OGSA.

The OGSI V1.0 specification proposes detailed specifications for the conventions that govern how clients create, discover, and interact with a grid service instance.

That is, it specifies

- How grid service instances are named and referenced;
- The base, common interfaces (and associated behaviors) that all grid services implement;
- The additional interfaces and behaviors associated with factories and service groups. The specification does not address how grid services are created, managed, and destroyed within any particular hosting environment.

The term hosting environment is used in the OGSI specification to denote the server in which one or more grid service implementations run. Such servers are typically language or platform specific; examples include native Unix and Window processes, J2EE application servers, and Microsoft .NET.
What is OGSA/OGSI? A More Detailed View

- **Setting the Context:**
  - GGF calls OGSI the “base for OGSA.”
  - Specifically, there is a relationship between OGSI and distributed object systems and also a relationship between OGSI and the existing (and evolving) Web services framework.
  - One needs to examine both the client-side programming patterns for grid services and a conceptual hosting environment for grid services.
  - The patterns described in this section are enabled but not required by OGSI.
What is OGSA/OGSI? A More Detailed View

- **Relationship to Distributed Object Systems:**
  - A given grid service implementation is an addressable and potentially stateful instance that implements one or more interfaces described by WSDL portTypes.
  - Grid service factories can be used to create instances implementing a given set of portType(s).
  - Each grid service instance has a notion of identity with respect to the other instances in the distributed grid.
  - Each instance can be characterized as state coupled with behavior published through type-specific operations.
  - The architecture also supports introspection in that a client application can ask a grid service instance to return information describing itself, such as the collection of portTypes that it implements.
  - Grid service instances are made accessible to (potentially remote) client applications through the use of a grid service handle and a grid service reference (GSR).
  - 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 portType(s) of the target service description directly to the specific instance at the specified network-attached service endpoint identified by the grid service reference.
• In many situations, client stubs and helper classes isolate application programmers from the details of using grid service references.
• Some client-side infrastructure software assumes responsibility for directing an operation to a specific instance that the GSR identifies.
• The characteristics introduced above (stateful instances, typed interfaces, global names, etc.) are frequently also cited as fundamental characteristics of distributed object-based systems.
• There are various other aspects of distributed object models (as traditionally defined) that are specifically not required or prescribed by OGSI.
• For this reason, OGSI does not adopt the term distributed object model or distributed object system when describing these concepts, but instead uses the term “open grid services infrastructure,” thus emphasizing the connections that are established with both Web services and grid technologies.
Among the object-related issues that are not addressed within OGSI are implementation inheritance, service instance 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 at the service provider side of the usage contract.

The grid service specification does not prescribe, dictate, or prevent the use of any particular development approach or hosting technology for grid service instances.

Grid service providers are free to implement the semantic contract of the service description in any technology and hosting architecture of their choosing.

OGSI envisions implementations in J2EE, .NET, traditional commercial transaction management servers, traditional procedural Unix servers, and so forth.

It also envisions service implementations in a wide variety of both object-oriented and non object-oriented programming languages.
• **Client-Side Programming Patterns:**
  - Another important issue is how OGSI interfaces are likely to be invoked from client applications.
  - OGSI exploits an important component of the Web services framework: the use of WSDL to describe multiple protocol bindings, encoding styles, messaging styles (RPC versus document oriented), and so on, for a given Web service.
  - The Web Services Invocation Framework (WSIF) and Java API for XML RPC (JAX-RPC) are among the many examples of infrastructure software that provide this capability.
The following figure depicts a possible (but not required) client-side architecture for OGSI.
What is OGSA/OGSI? A More Detailed View

- In this approach, a clear separation exists between the client application and the client-side representation of the Web service (proxy), including components for marshaling the invocation of a Web service over a chosen binding.
- In particular, the client application is insulated from the details of the Web service invocation by a higher-level abstraction: the client-side interface.
- Various tools can take the WSDL description of the Web service and generate interface definitions in a wide range of programming-language-specific constructs (e.g., Java interfaces and C#).
- This interface is a front end to specific parameter marshaling and message routing that can incorporate various binding options provided by the WSDL.
- Further, this approach allows certain efficiencies, for example, detecting that the client and the Web service exist on the same network host, therefore avoiding the overhead of preparing for and executing the invocation using network protocols.
What is OGSA/OGSI? A More Detailed View

- Within the client application runtime, a proxy provides a client-side representation of remote service instance’s interface.
- Proxy behaviors specific to a particular encoding and network protocol (binding, in Web services terminology) are encapsulated in a protocol-specific (binding-specific) stub.
- Details related to the binding specific access to the grid service instance, such as correct formatting and authentication mechanics, happen here; thus, the application is not required to handle these details itself.
- It is possible, but not recommended, for developers to build customized code that directly couples client applications to fixed bindings of a particular grid service instance.
- Although certain circumstances demand potential efficiencies gained by this style of customization, this approach introduces significant inflexibility into a system and therefore should only be used under extraordinary circumstances.
The developers of the OGSI specification expect the stub and client-side infrastructure model that we describe to be a common approach to enabling client access to grid services. This includes both application-specific services and common infrastructure services that are defined by OGSA. Thus, for most software developers using grid services, the infrastructure and application-level services appear in the form of a class library or programming language interface that is natural to the caller.

WSDL and the GWSDL extensions provide support for enabling heterogeneous tools and enabling infrastructure software.
What is OGSA/OGSI? A More Detailed View

- **Client Use of Grid Service Handles and References:**
  - As noted, a client gains access to a grid service instance through grid service handles and grid service references.
  - A grid service handle (GSH) can be thought of as a permanent network pointer to a particular grid service instance.
  - The GSH does not provide sufficient information to allow a client to access the service instance; the client needs to “resolve” a GSH into a grid service reference (GSR).
  - The GSR contains all the necessary information to access the service instance.
  - The GSR is not a “permanent” network pointer to the grid service instance because a GSR may become invalid for various reasons; for example, the grid service instance may be moved to a different server.
  - OGSI provides a mechanism, the HandleResolver to support client resolution of a grid service handle into a grid service reference.
What is OGSA/OGSI? A More Detailed View

Client Application

GSR

Resolve this GSH

HandleResolver Grid Service

Save

cache

Handle Resolver Specific resolver protocol

HandleResolver Grid Service

Resolving a GSH.
What is OGSA/OGSI? A More Detailed View

- The client resolves a GSH into a GSR by invoking a HandleResolver grid service instance identified by some out-of-band mechanism.
- The HandleResolver can use various means to do the resolution; some of these means are depicted in the previous figure.
- The HandleResolver may have the GSR stored in a local cache.
- The HandleResolver may need to invoke another HandleResolver to resolve the GSH.
- The HandleResolver may use a handle resolution protocol, specified by the particular kind (or scheme) of the GSH to resolve to a GSR.
- The HandleResolver protocol is specific to the kind of GSH being resolved.
- For example, one kind of handle may suggest the use of HTTP GET to a URL encoded in the GSH in order to resolve to a GSR.
What is OGSA/OGSI? A More Detailed View

• Relationship to Hosting Environment:
  – OGSI does not dictate a particular service-provider-side implementation architecture.
  – A variety of approaches are possible, ranging from implementing the grid service instance directly as an operating system process to a sophisticated server-side component model such as J2EE.
  – In the former case, most or even all support for standard grid service behaviors (invocation, lifetime management, registration, etc.) is encapsulated within the user process; for example, via linking with a standard library.
  – In the latter case, many of these behaviors are supported by the hosting environment.
Two approaches to the implementation of argument demarshaling functions in a grid service hosting environment.
What is OGSA/OGSI? A More Detailed View

- The previous figure illustrates **these differences** by showing two different approaches to the implementation of argument demarshaling functions.
- **One can assume that** the invocation message **is received at a network protocol termination point** (e.g., an HTTP servlet engine) that **converts the data in the invocation message into a format consumable by the hosting environment.**
- **The top part of Figure illustrates** two grid service instances (the oval) associated with container-managed components (e.g., EJBs within a J2EE container).
- **Here, the message is dispatched to these components, with the container frequently providing facilities for demarshaling and decoding the incoming message from a format** (such as an XML/SOAP message) **into an invocation of the component in native programming language.**
- **In some circumstances (the oval), the entire behavior of a grid service instance is completely encapsulated within the component.**
What is OGSA/OGSI? A More Detailed View

- In other cases (the oval), a component will collaborate with other server-side executables, perhaps through an adapter layer, to complete the implementation of the grid service behavior.
- The bottom part of figure depicts another scenario wherein the entire behavior of the grid service instance, including the demarshaling/decoding of the network message, has been encapsulated within a single executable.
- Although this approach may have some efficiency advantages, it provides little opportunity for reuse of functionality between grid service implementations.
- A container implementation may provide a range of functionality beyond simple argument demarshaling.
- For example, the container implementation may provide lifetime management functions, automatic support for authorization and authentication, request logging, intercepting lifetime management functions, and terminating service instances when a service lifetime expires or an explicit destruction request is received. Thus, one avoids the need to reimplement these common behaviors in different grid service implementations.
Grid Services

• The purpose of the OGSI document is to specify the (standardized) interfaces and behaviors that define a grid service.

• In brief, a grid service is a WSDL-defined service that conforms to a set of conventions relating to its interface definitions and behaviors. Thus, every grid service is a Web service, though the converse of this statement is not true.

• The OGSI document expands upon this brief statement by:
  – Introducing a set of WSDL conventions that one uses in the grid service specification; these conventions have been incorporated in WSDL 1.2.
  – Defining service data that provide a standard way for representing and querying metadata and state data from a service instance.
  – Introducing a series of core properties of grid service, including:
    • Defining grid service description and grid service instance, as organizing principles for their extension and their use.
    • Defining how OGSI models time
    • Defining the grid service handle and grid service reference constructs that are used to refer to grid service instances.
    • Defining a common approach for conveying fault information from operations. This approach defines a base XML schema definition and associated semantics for WSDL fault messages to support a common interpretation; the approach simply defines the base format for fault messages, without modifying the WSDL fault message model.
    • Defining the life cycle of a grid service instance
Grid Services

• **WSDL Extensions and Conventions:**
  – OGSI is based on Web services; in particular, it uses WSDL as the mechanism to describe the public interfaces of grid services.
  – WSDL 1.1 is deficient in two critical areas:
    • lack of interface (portType) extension and
    • the inability to describe additional information elements on a portType (lack of open content).

• These deficiencies have been addressed by the W3C Web Services Description Working Group.
• Because WSDL 1.2 is a “work in progress,” OGSI cannot directly incorporate the entire WSDL 1.2 body of work.
• Instead, OGSI defines an extension to WSDL 1.1, isolated to the `wsdl:portType` element, which provides the minimal required extensions to WSDL 1.1.
• These extensions to WSDL 1.1 match equivalent functionality agreed to by the W3C Web Services Description Working Group.
• Once WSDL 1.2 is published as a recommendation by the W3C, the Global Grid Forum is committed to defining a follow on version of OGSI that exploits WSDL 1.2, and to defining a translation from this OGSI v1.0 extension to WSDL 1.2.
• **Service Data:**
  – The approach to stateful Web services introduced in OGSI identified the need for a common mechanism to expose a service instance’s state data to service requestors for query, update, and change notification.
  – Since this concept is applicable to any Web service including those used outside the context of grid applications, one can propose a common approach to exposing Web service state data called “serviceData.”
  – The GGF is endeavoring to introduce this concept to the broader Web services community.
In order to provide a complete description of the interface of a stateful Web service (i.e., a grid service), it is necessary to describe the elements of its state that are externally observable.

By externally observable, one means that the state of the service instance is exposed to clients making use of the declared service interface, where those clients are outside of what would be considered the internal implementation of the service instance itself.

The need to declare service data as part of the service’s external interface is roughly equivalent to the idea of declaring attributes as part of an object-oriented interface described in an object-oriented interface-definition language.
• Service data can be exposed for read, update, or subscription purposes.
• Since WSDL defines operations and messages for portTypes, the declared state of a service must be externally accessed only through service operations defined as part of the service interface.
• To avoid the need to define serviceData-specific operations for each serviceData element, the grid service portType provides base operations for manipulating serviceData elements by name.
• Consider an example. Interface alpha introduces operations op1, op2, and op3. Also assume that the alpha interface consists of publicly accessible data elements of de1, de2, and de3. One uses WSDL to describe alpha and its operations. The OGSI serviceData construct extends WSDL so that the designer can further define the interface to alpha by declaring the public accessibility of certain parts of its state de1, de2, and de3. This declaration then facilitates the execution of operations on the service data of a stateful service instance implementing the alpha interface.
• Put simply, the serviceData declaration is the mechanism used to express the elements of the publicly available state exposed by the service’s interface. ServiceData elements are accessible through operations of the service interfaces such as those defined in this specification. The private internal state of the service instance is not part of the service interface and is therefore not represented through a serviceData declaration.