From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution

Version 1.1

3/05/2004

Authors

Karl Czajkowski (Globus Alliance / USC Information Sciences Institute)

Don Ferguson (IBM)

Ian Foster (Globus Alliance / Argonne National Laboratory)

Jeff Frey (IBM)

Steve Graham (IBM)

Tom Maguire (IBM)

David Snelling (Fujitsu Laboratories of Europe)

Steve Tuecke (Globus Alliance / Argonne National Laboratory)

Abstract

The Open Grid Services Infrastructure specification version 1.0 (OGSI), released in July 2003, defines a set of conventions and extensions on the use of Web Service Definition Language and XML Schema to enable stateful Web services. It introduces the idea of a stateful Web services and defines approaches for creating, naming, and managing the lifetime of instances of services; for declaring and inspecting service state data; for asynchronous notification of service state change; for representing and managing collections of service instances; and for common handling of service invocation faults. In January 2004, the WS-Resource Framework was proposed as a refactoring and evolution of OGSI aimed at exploiting new Web services standards, specifically WS-Addressing, and at evolving OGSI based on early implementation and application experiences. The WS-Resource Framework retains essentially all of the functional capabilities present in OGSI, while changing some of the syntax (e.g., to exploit WS-Addressing) and also adopting a different terminology in its presentation. In addition, the WS-Resource Framework partitions OGSI functionality into five distinct, composable specifications. In this document, we explain the relationship between OGSI and the WS-Resource Framework and the related WS-Notification family of specifications, explain the common requirements that both address, and compare and contrast the approaches taken to the realization of those requirements.

Copyright Notice

Table of Contents

1 Introduction

The Open Grid Services Infrastructure specification version 1.0 (OGSI) [OGSI-Spec], released in July 2003 by the OGSI Working Group of the Global Grid Forum, defines a set of conventions and extensions on the use of Web Service Definition Language (WSDL) and XML Schema to enable stateful Web services. It defines approaches for creating, naming, and managing the lifetime of instances of services; for declaring and inspecting service state data; for asynchronous notification of service state change; for representing and managing collections of service instances; and for common handling of service invocation faults.

At the core of OGSI is a *Grid service* [Physiology], a Web service that conforms to a set of conventions for such purposes as service lifetime management, inspection, and notification of service state changes. Grid services 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, and a base fault type.

In parallel with and subsequent to this OGSI work, the Web services architecture has evolved, with for example the definition of WSDL 2.0 [WSDL 2.0] progressing and the release of new draft specifications such as WS-Addressing [WS-Addressing]. These developments make it timely to consider how the functional capabilities of OGSI exploit functionality provided by other specifications (in particular, WS-Addressing) and to align OGSI functions with the emerging consensus on Web services architecture [WS-Arch]. OGSI 1.0 also combined into one specification functions that are independently useful, for example event notification. It is appropriate to factor the OGSI interfaces to produce a framework of independently useful Web service standards.

A recent effort aimed at achieved such a refactoring has produced the specifications listed in Table 1, five of which are named collectively the *WS-Resource Framework* [State Paper], while the WS-Notification family of specifications [WS-Notification] addresses notification (event) subscription and delivery. Collectively, these specifications retain all of the essential *functional capabilities* present in OGSI, while changing some of the *syntax* (e.g., to exploit WS-Addressing) and adopting a different *terminology* in its presentation. In addition, the specifications partition OGSI functionality into distinct functionality that allows flexible composition in a mix-and-match manner. The factoring, composition capability and greater reliance on broadly accepted Web service concepts provide a simpler, more familiar and incremental path for developers wishing to exploit OGSI functionality.

The purpose of this paper is to explain how the new WS-Resource Framework and WS-Notification specifications derive from and relate to the OGSI specification. To this end, we explain how each OGSI constructs realization in the new specifications, and point out areas in which the new specifications provide different or enhanced functionality. Our goals in defining this mapping from OGSI are to:

1. persuade the OGSI community that the new specifications are a useful refactoring and evolution of OGSI;

2. make clear the intellectual debts that the new specifications owe to OGSI and in doing so also highlight the architectural differences between the new specifications and OGSI; and
3. Summarize the issues that arise in migrating OGSI-based applications to the WS-Resource Framework and WS-Notification specifications.

In the rest of this paper, we first review OGSI and related Web services specifications (Section 2) and introduce the principal concepts that underlie the WS-Resource Framework, including the WS-Resource construct (Section 3). Then, we compare and contrast in turn the OGSI and the WS-Resource Framework treatments of service addressing (Section 4), resource properties (Section 5), lifetime management (Section 6), service groups (Section 7), and faults (Section 8), and the WS-Notification treatment of notification (Section 9). Finally, we discuss issues that arise when migrating applications from OGSI to WS-Resource Framework and WS-Notification (Section 10), and conclude. We have tried to make this paper accessible to the reader unfamiliar with OGSI and WS-Resource Framework, but the reader who wishes to understand technical details will need to read the relevant technical specifications.

Table 1: The refactoring of OGSI yields five normative WS-Resource Framework specifications plus WS-Notification

Name	Description
WS-ResourceProperties	Describes associating stateful resources and Web services to produce WS-Resources, and how elements of publicly visible properties of a WS-Resource are, retrieved, changed, and deleted.
WS-ResourceLifetime	Allow a requestor to destroy a WS-Resource either immediately or at a scheduled future point in time.
WS-RenewableReferences	Annotate a WS-Addressing endpoint reference with information needed to retrieve a new endpoint reference when the current reference becomes invalid.
WS-ServiceGroup	Create and use heterogeneous by-reference collections of Web services.
WS-BaseFault	Describes a base fault type used for reporting errors.
WS-Notification family of specifications	Standard approaches to notification using a topic-based publish and subscribe pattern.

2 Background

We provide some background on OGSI and relevant Web services specifications.

2.1 Open Grid Services Infrastructure

OGSI is concerned primarily with creating, addressing, inspecting, and managing the lifetime of stateful *Grid services* [Physiology]. The OGSI version 1.0 specification [OGSI-Spec], released in July 2003, defines a Grid service to be 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 advanced distributed applications.

OGSI defines a component model by using extended WSDL and XML Schema definition to introduce the concepts of stateful Web service instances, common metadata and inspection, asynchronous notification of state change, references to instances of services, collections of service instances, and service state data declaration that augments the constraint capabilities of XML Schema definition. More specifically, the OGSI specification defines:

- A set of WSDL extensions some of which have analogous support in WSDL 2.0 [WSDL 2.0 DRAFT].
- WSDL constructs and standard operations for representing, querying, and updating *service data* (metadata and state data) associated with a service.
- The Grid Service Handle and Grid Service Reference constructs, used to address Grid services.
- A definition of common fault information from operations that defines a base XML Schema 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.
- A set of operations for creating and destroying Grid services that provides for both explicit destruction of services and implicit garbage collection of expired services without the need for explicit destruction.
- A set of operations for creating and using heterogeneous by-reference collections of Web services.
- Mechanisms for requesting asynchronous notifications of changes in the value of service data elements.

At least six different implementations of the OGSI specification exist and some early experience has been gained with the use of OGSI constructs in applications. In addition, various efforts have started to develop higher-level specifications that build on OGSI constructs.

2.2 Web Services Architecture Evolution

Since development started on OGSI in early 2002, the Web Services world has evolved significantly. Specifically, a number of new specifications and use patterns

have emerged that simplify and clarify the ideas expressed in OGSI. The following briefly outlines this evolution.

WS-Addressing provides transport-neutral mechanisms to address Web services. Specifically, WS-Addressing specification defines XML elements to identify Web service endpoints and to include endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner. The end point reference information provides not only the address of Web service itself, but can also serve to identify stateful resources "behind" the service by using endpoint reference properties.

Although less central to the WS-Resource definition, WS-MetaDataExchange provides a collection of mechanisms for obtaining information about a published service, such as its WSDL description, XML Schema definitions and any other policy information necessary to use the service.

Since WS-Addressing and WS-MetaDataExchange provide several capabilities that are also defined within OGSI, it is beneficial to exploit those Web services specifications rather than maintaining a specification that defines the same functionality redundantly.

2.3 Web Services Critiques of OGSI

While the motivation for the WS-Resource Framework is primarily the desire to integrate recent developments in Web services architecture, in particular WS-Addressing, its design also addresses four criticisms of OGSI from the Web services community.

1. *Too much stuff in one specification*. OGSI did not have a clean separation (factoring) of functions to support incremental adoption. For example, event notification is a useful function independent of coupling with service data. Metadata introspection is a useful concept that does not require expression through service data. The WS-Resource Framework and WS-Notification specifications address this critique by partitioning OGSI v1.0 functionality into a family of separate specifications that allow for flexible composition.
2. *Does not work well with existing Web services and XML tooling*. OGSI v1.0 uses XML Schema aggressively, for example with substantial use of xsd:any, attributes, etc., and "document-oriented" WSDL operations. These features cause problems with, for example, JAX-RPC. The WS-Resource Framework uses standard XML Schema mechanisms that are familiar to developers and are supported by existing tooling. Instead of extending the WSDL 1.1 portType definition, the WS-Resource Framework defines a means to annotate the WSDL 1.1 portType to associate this XML information model of the resource with Web service operations. This annotation is a legal construct within the WSDL 1.1 language.
3. *Too object oriented*. OGSI v1.0 models a stateful resource as a Web service that encapsulates the resource's state, with the identity and lifecycle of the service and resource state coupled. This approach has spurred anxiety among some Web services purists who argue, "Web services do not have state or instances". In addition, some Web services implementations do not

accommodate dynamic service creation and destruction. The WS-Resource Framework re-articulates the underlying OGSI architecture to make an explicit distinction between the "service" and the stateful entities acted upon by that service. The WS-Resource Framework defines the means by which a Web service and a stateful resource are composed. The WS-Resource Framework calls these compositions "WS-Resources," and introduces the "implied resource pattern" to formalize the relationship between Web services and the stateful resources through a conventional use of WS-Addressing.

4. *Introduction of forthcoming WSDL 2.0 capability as unsupported extensions to WSDL 1.1.* The OGSI authors exploited constructs from the proposed WSDL 2.0 draft specification. Delays in the publication of WSDL 2.0 made it more difficult to support the OGSI definition with existing Web services tooling and runtimes. Therefore, it is beneficial to express the capabilities of OGSI using the WSDL 1.1 definition to avoid the requirement for extended tooling.

3 From OGSI to the WS-Resource Framework

We introduce the general approach and underlying motivations for the factoring and evolution process that takes OGSI to the WS-Resource Framework. This factoring is done in terms of three evolutionary steps:

- the introduction of the WS-Resource concept;
- better separation of function and exploitation of other Web services specifications; and
- A broader view of notification, which is a general Web service requirement upon which state change notification can be built.

We provide first an overview of the modifications made when moving from OGSI to the WS-Resource Framework and then present details in subsequent sections. The relevant technical specifications (Table 1) can be consulted for WS-Resource Framework details. Table 2 summarizes the mappings from OGSI concepts and constructs to equivalent WS-Resource Framework concepts and constructs.

3.1 The WS-Resource Construct

The WS-Resource Framework is concerned primarily with the creation, addressing, inspection, and lifetime management of stateful resources. The framework provides the means to express state as stateful resources and codifies the relationship between Web services and stateful resources in terms of the *implied resource pattern*, which is a set of conventions on Web services technologies, particularly XML, WSDL, and WS-Addressing [WS-Addressing]. The composition of a stateful resource and a Web service that participates in the implied resource pattern is termed a *WS-Resource.* The framework describes the WS-Resource definition, and describes how to make the properties of a WS-Resource accessible through a Web service interface, and to manage and reason about a WS-Resource's lifetime.

Table 2: How the primary OGSI constructs map to WS-Resource Framework and WS-Notification constructs

OGSI	WS-Resource Framework	Comments
Grid Service Reference	WS-Addressing Endpoint Reference	Uses the endpoint reference properties of WS-Addressing to identify a stateful resource associated with the Web service at the designated endpoint.
Grid Service Handle	WS-Addressing Endpoint Reference & WS-RenewableReferences	WS-RenewableReferences introduces policy annotations to the WS-Addressing endpoint reference that allow "handles" and "handle Resolvers" to be an integrated part of the endpoint reference. Use of the policy annotations provides for additional endpoint reference stability.
HandleResolver portType	WS-RenewableReferences	Integration of Handle Resolution service references in the endpoint reference.
Service data definition	Resource properties definition	Better exploits XML Schema. Compatible with WSDL 1.1. Removes modifiability and mutability metadata.
GridService portType service data access	WS-ResourceProperties	Multiple operations instead of one extensible operation, supporting simpler binding to existing programming models.
GridService portType lifetime management	WS-ResourceLifetime	Removes the superfluous "terminate before" operation. Cosmetic changes to others.
Notification portTypes	WS-Notification	Generalizes notification to hierarchical topic-based pub/sub mechanism.
Factory portType		Now treated as a pattern; thus, no specific operation.
ServiceGroup portTypes	WS-ServiceGroup	Only minor changes
Base fault type	WS-BaseFault	Only minor changes.
GWSDL	Cut-and-paste	Use existing WSDL 1.1 interface composition approaches (i.e., cut-and-paste). Wait for WSDL 2.0 adoption to enable support for Web service tooling and runtimes.

As this brief description suggests, both OGSI and the WS-Resource Framework are concerned with how to manipulate stateful resources through a Web services interface. Furthermore, while the two approaches model stateful resources differently—as a Grid service and a WS-Resource, respectively—they provide essentially equivalent functionality and use semantically similar WSDL interface definitions. Both Grid services and WS-Resources can be created, addressed, inspected, and destroyed, and in essentially the same ways.

The WS-Resource Framework has two advantages relative to OGSI. First, it better exploits existing XML standards, as well as emerging Web services standards such as WS-Addressing. Thus, the WS-Resource Framework is easier to implement within existing and emerging Web services toolkits, and easier to exploit within the myriad of Web services interfaces in definition. The second advantage is pedagogical. OGSI terminology and constructs have caused anxiety for some in the Web services community due to a mistaken view that OGSI implies that Web services must become heavyweight constructs. The WS-Resource Framework makes it clear that this is not the intention or consequence: the goal is simply to allow stateful resources manipulation via Web services operations. Nothing in either the OGSI or the WS-Resource Framework model prevents a Web services implementation from being a stateless message processor that dispatches operations to backend resources. The WS-Resource Framework model makes this fact clearer, due to its more direct translation to an implementation approach that separates message processors from stateful resources.

3.2 Other Changes

The WS-Resource Framework also incorporates changes motivated by other lessons learned since the completion of the OGSI 1.0 specification. We summarize some of the more notable changes here and provide a more detailed description in subsequent sections.

Implementation experience shows that the OGSI Factory interface provides little useful functionality. Thus, the WS-Resource Framework defines instead the more general *WS-Resource factory pattern*. Even within OGSI, there are several uses of a factory pattern where, for clarity of expression and type control, explicit operations exist rather than relying on the generic 'create' operation in the Factory portType, e.g. the 'add' operation in ServiceGroupRegistration portType.

The OGSI Notification interfaces do not support a variety of functions required in a general eventing system and supported by existing message-oriented middleware. The family of WS-Notification specifications was created to address this gap.

OGSI uses the Grid Service Reference (GSR) as an address for a Grid service and introduces the OGSI Grid Service Handle (GSH) construct and HandleResolver mechanism as one (underspecified) way of handling mappings between abstract, long-lived names and concrete, perhaps short-lived addresses. The combination of these three OGSI constructs provides several distinct functions in an interdependent collection of mechanisms. The WS-Resource Framework defines a framework for these functions and provides the independent mechanisms. The WS-RenewableReferences specification defines the ability to make endpoint references stable references by the addition of endpoint policy assertions.

The large size and scope of the OGSI specification has made it hard for readers to understand its contents and to identify and refer to those components that are required for a specific task. Thus, the WS-Resource Framework partitions OGSI functionality into distinct specifications that allow flexible composition.

OGSI uses XML Schema aggressively and in particular makes substantial use of extensibility (e.g., xsd:any). Unfortunately, this use of these standard XML Schema features has caused problems with some existing Web services toolkits, XML development tools, and standards (e.g., JAX-RPC). Thus, the WS-Resource Framework takes a somewhat more conservative approach to the use of XML Schema, for example by using multiple operations in place of a single, extensible operation as in OGSI.

OGSI's GWSDL extension to the WSDL 1.1 portType is mainly syntactic sugar, to allow for interface extension. In addition, GWSDL went beyond syntactic sugar with the declaration of service data as part of an interface definition. Unfortunately, GWSDL was a major barrier to the use of OGSI. Thus, the WS-Resource Framework simply defines its messages in terms of WSDL 1.1, and requires that designers of composite interfaces copy-and-paste together the components of such an interface until WSDL 2.0 is completed.

4 Stateful Resource Addressing

We now proceed to discuss the WS-Resource Framework rendering of each OGSI construct in turn, presenting first requirements and then comparing and contrasting the OGSI and WS-Resource Framework approaches to meeting those requirements. We start with addressing.

Because the WS-Resources to which we wish provide access via service-oriented mechanisms are dynamic and stateful, we need to be able to distinguish one dynamically created WS-Resource from another, and provide the means to address these WS-Resources reliably across a Web services infrastructure in a convenient and interoperable way.

A minimum requirement for a network-wide WS-Resource addressing construct is that it must standardize the representation of the address of the associated Web service deployed at a given network endpoint. In addition to the endpoint address of the Web service, the addressing construct may contain other metadata associated with the Web service such as service description information and reference properties associated to a contextual use of the targeted Web service.

Typically, an authoritative source provides Web service addressing constructs (Web service "endpoint references") and associated policy information. The endpoint reference made available to a client represents a copy of the addressing and policy information that may, at some point, become incoherent due to changes introduced by the authoritative source that effects the endpoint location and/or the policy assertions governing message exchanges with the Web service. Mechanisms that allow the Web service endpoint references to be "renewed" in the event they become invalid would provide additional stability in the addressing scheme.

OGSI addresses these requirements by defining the Grid Service Handle (GSH) and the Grid Service Reference (GSR) constructs. The GSH does not provide sufficient addressing information to allow a client to access the service instance, but it is a

more stable "virtualized" expression of the service "endpoint reference"; the client needs to "resolve" a GSH into a GSR, which contains the necessary information in order to communicate with the stateful Web service instance. OGSI provides a mechanism, the HandleResolver to support client resolution of a GSH into a GSR. The HandleResolver portType defines a standard means for resolving a GSH to a GSR, independent of any particular GSH scheme. We refer to a service instance that implements the HandleResolver portType as a handle resolver.

In contrast, the WS-Resource Framework builds on the recently published WS-Addressing specification to achieve the same goals in slightly different ways. First, it adopts the endpoint reference construct defined in the WS-Addressing specification as an XML syntax for identifying Web service endpoints. It then defines a particular usage pattern for endpoint references, the *implied resource pattern*, in which the *reference properties* field of the endpoint reference contains an identifier of a specific stateful resource associated with the Web service. These two pieces of information are the logical equivalent of the addressing content of the OGSI defined GSR.

Second, rather than distinguishing between two fixed types of names, immutable GSHs and potentially mutable GSRs, it introduces (in WS-RenewableReferences) a mechanism for associating with any endpoint reference (not just one that refers to a WS-Resource) a "resolver service." Specifically, WS-RenewableReferences allows a renewable reference policy to be associated with an endpoint reference. This WS-Policy statement can include an assertion concerning the ReferenceResolver for obtaining a new reference for a particular service.

The quality of naming of OGSI services and WS-Resources provided by OGSI and the WS-Resource Framework, respectively, are equivalent. There may be multiple OGSI GSHs to the same service, while in the WS-Resource Framework there may be multiple endpoint references for the same resource. Two GSHs can only be compared for equality via syntactic comparison, but service inequality cannot be deduced from GSH syntactic inequality. The same is true of endpoint references. Finally, non-reuse of a GSH is guaranteed; that is, the same GSH will never refer to a different service. A WS-Resource qualified endpoint reference provides the same guarantee. It is expected that the quality of identity will be enhanced for specific use cases such as resource and service management [WSDM].

One small feature of an OGSI GSH is its URI syntax, thus making it a short, human-readable "name" for a service. There is not equivalent feature in the WS-Resource Framework. Instead, various forms of naming services can be built on top of the WS-Resource Framework, which can provide whatever from of name desired, and which map to endpoint references.

Thus, the WS-Resource Framework provides virtually all functionality present in OGSI, and has the advantages of leveraging WS-Addressing, allowing for arbitrary hierarchies of resolver services, and allowing to be used independently of each other.

5 Resource Properties

The second set of requirements concerns mechanisms for defining the message exchanges used to access the state of a stateful entity. More specifically, we require the ability to

1) determine the type of the state and thus the specific message exchanges that may be supported, and
2) issue read, modify, and query requests against state components.

Both OGSI and the WS-Resource Framework take essentially the same approach to addressing these two requirements, but use different syntax.

OGSI meets the first requirement by declaring service data elements as part of an interface definition. When multiple interfaces are composed using the GWSDL (or equivalently, WSDL 2.0) interface extension, the service data element declarations are implicitly aggregated to create the complete service data set.

The WS-Resource Framework uses standard XML Schema global element declarations to define resource property elements. A Resource properties document collects resource property elements and the resource properties document is associated with a Web service interface by using an XML attribute on the WSDL 1.1 portType. In this way the existence and type of a resource properties document is captured, as well as its association with a particular portType. The annotated portType defines the overall type of the WS-Resource. This construction is legal WSDL 1.1, thanks to the revised WSDL 1.1 schema required by WS-Interoperability Basic Profile 1.0. However, when combining messages from multiple interfaces into a single interface via copy-and-paste, it is necessary to combine the resource property elements from the various interfaces into a single resource property document via copy-and-paste as well.

Resource property elements are almost identical to service data elements. The only difference is that resource property element declarations are simply XML global element declarations, whereas OGSI service data element declarations use an OGSI-specific syntax that mirrors an XML element declaration. A result is that element declarations for resource properties cannot contain annotations of modifiability and mutability attributes, as can be done in OGSI. These attributes, if deemed critical for some applications, could be defined in some other manner such as metadata attachments, but the WS-Resource Framework has not defined any such approach.

OGSI meets the second requirement via a small set of extensible operations, in particular findServiceData and setServiceData, with a required extension support multi-element get/set. The WS-Resource Framework instead introduces, in WS-ResourceProperties, a set of more specific operations for getting and setting resource properties: single element get, multi-element get/set, and XPath query. Others may define additional operations as desired. Thus, thanks to the XPath query in WS-ResourceProperties, the functionality provided by the WS-Resource Framework for accessing resource property elements is a superset of that provided by OGSI.

6 Lifetime Management

The lifetime of a stateful entity is defined to be the period between its creation and its destruction. The actual mechanisms by which a particular stateful entity is created and destroyed are implementation-specific, and therefore not defined or prescribed in either OGSI or the WS-Resource Framework. However, we do need to address three aspects of the entity lifecycle: creation, identity assignment, and destruction. Both OGSI and the WS-Resource Framework address these three issues in

essentially the same way. The mapping from OGSI to WS-Resource Framework constructs is summarized in Table 3 and described in the following.

OGSI addresses the idea of service creation via the Factory portType, which provides an operation "createService" that takes as optional arguments a proposed termination time and execution parameters, and returns (upon success) an OGSI defined service locator for the newly created service, an initial termination time, and optional additional data. In practice, the standardization of this operation provided only limited value, as most parameters provided to a particular Factory implementation would be implementation-specific.

For these reasons, the WS-Resource Framework defines simply the *factory pattern*, a term used to denote a Web service that supports an operation that creates, and returns endpoint references for, one or more new WS-Resources [WS-ResourceLifetime]. The WS-Resource Framework factory pattern can provide the same functionality as the OGSI factory operation. Recall that the OGSI definition of a stateful Web service is now a WS-Resource. Thus, the creation of a stateful Web service in OGSI terms is really the creation of a WS-Resource in WS-Resource Framework terms.

Table 3: Mapping from OGSI to WS-Resource Framework lifetime management constructs

Function	OGSI	WS-Resource Framework
Create new entity	Factory portType operation "createService"	Factory pattern definition
Address the entity	Grid Service Reference and Grid Service Handle	WS-Addressing Endpoint Reference with reference properties.
Immediate destruction	GridService portType operation "destroy."	ResourceLifetime portType operation "Destroy." However, this operation is synchronous in WS-Resource Framework.
Scheduled destruction	GridService portType operations "requestTerminationAfter" and "requestTerminationBefore"	ResourceLifetime portType operation "SetTerminationTime" is equivalent to "after." "Before" was determined to be superfluous in the absence of real-time scheduling.
Determine current time	GridService portType service data element "CurrentTime"	Resource property "CurrentTime"
Determine lifetime	GridService portType service data element "terminationTime"	Resource property "TerminationTime"
Notify of destruction	Not available	Subscribe to topic "ResourceDestruction"

A requestor that requests a factory to create a new stateful entity will typically only be interested in that new entity for some finite period. After that time, it should be possible to destroy the new entity so that associated system resources can be reclaimed. Two destruction methods are of interest: *immediate destruction*, in which the requestor sends a destroy request, and *scheduled* (also known as *soft-state*) *destruction*, in which an entity has an assigned lifetime after which the entity can be destroyed by the resource provider. By resource provider, we mean any component in the system responsible for hosting the implementation of the resource.

OGSI addresses destruction via operations supported in its GridService portType. The Destroy operation allows a requestor to request destruction of a Grid service, while the requestTerminationAfter and requestTerminationBefore operations allow a requestor to manage a Grid service's lifetime.

The WS-Resource Framework WS-ResourceLifetime specification defines equivalent message exchanges. A service requestor that wishes to destroy a WS-Resource explicitly must use the appropriate WS-Resource-qualified endpoint reference to send a destroy request message to the WS-Resource. The reference properties within the endpoint reference identify the stateful resource component of the WS-Resource targeted for destruction. Note that the destruction of the stateful resource component of a WS-Resource effectively destroys the WS-Resource. Unlike in OGSI, a successful response from a destroy operation indicates that the resource has been destroyed and can no longer be accessed via that service. A successful return in OGSI only indicates that destruction has been initiated.

The WS-Resource Framework defined SetTerminationTime operation supports scheduled destruction in the same way as the OGSI defined "requestTerminationAfter"; no equivalent to "requestTerminationBefore" is provided as that operation is superfluous in the absence of real-time scheduling.

A final requirement relating to scheduled destruction is that a requestor may need to be able to determine the stateful resource's view of the current time and its associated termination time. OGSI and the WS-Resource Framework address these requirements in the same manner: via two service data elements (OGSI) or resource properties (WS-Resource Framework): CurrentTime and TerminationTime.

7 Service Groups

The term *service group* refers to a standard mechanism for creating a heterogeneous by-reference collection of Web services. Service groups can form a wide variety of collections of services, including building registries of services. While their use is not restricted to Grid services (in OGSI) or WS-Resources (in the WS-Resource Framework), service groups are particularly important when dealing with stateful entities.

OGSI and the WS-Resource Framework address this requirement in essentially the same way, via the OGSI ServiceGroup portTypes and the equivalent interfaces defined in the WS-ServiceGroup specification, respectively. The only difference between the two approaches is that WS-ServiceGroup removes the "remove" operation on the ServiceGroupRegistration interface, which allowed for removal of a set of matching services. This operation was removed mainly because of the open

extensibility of this operation, and its redundancy with removing services from a group by doing lifetime management on the service group entry resource.

8 Faults

WSDL defines a message exchange fault model, but not a base format for fault messages. Specific domains that define interfaces using WSDL would typically define a "fault schema" for utilization across various message exchanges defined in those interfaces. Both OGSI and the WS-Resource Framework define interfaces using WSDL, and without a common base fault mechanism there is no basis for a common interpretation of fault messages generated by different sources. Interoperability requires the common interpretation.

OGSI addresses this issue by defining a base XML Schema definition (a base XSD type, ogsi:FaultType) and associated semantics for fault messages, along with a convention for extending this base definition for various types of faults. This definition simplifies problem determination by having a common base set of information that all fault messages contain. Note that the approach simply defines the base format for fault messages, without modifying the WSDL fault message model.

The WS-Resource Framework adopts the same constructs, defining them in the WS-BaseFault specification. The only difference is the removal of the open extensibility from WS-BaseFault, because it is redundant with the required approach of extending the base fault type using XML Schema extension for extended faults, and that extensibility element placed an additional burden upon the capabilities of broadly available Web services tooling.

9 Notification

In an environment in which stateful resources may be created and destroyed, and may change their state, dynamically, it becomes important to provide support for asynchronous notification of changes in the state of individual resources and/or other system components such as registries.

OGSI meets this requirement via its Notification portTypes, which allow a client to define a subscription (a persistent query) against one or more service data values.

Subscription and notification is a broad concept. Not all "events" relate to changes in the "state" of a service or resource. The WS-Notification family of specifications introduce a more feature-complete, generic, hierarchical topic-based approach for publish/subscribe-based notification, which is a common model followed in large-scale, distributed event management systems. WS-Resource Properties then defines a mapping from element names of resource properties to topic names to support functionality similar to OGSI service data subscription. WS-Notification includes richer support for controlling subscriptions (e.g., pause and resume), and for defining intermediaries.

10 Porting Interfaces

The porting of interfaces from OGSI to the WS-Resource Framework and WS-Notification is straightforward and comprises a set of mechanical transformations.

WSDL operation definitions themselves need not change. What changes is simply how we talk about the operations: we must do so in terms of the WS-Resource model (i.e., in terms of operations on WS-Resources), rather than the OGSI Grid service model.

The removal of GWSDL means that we must instead copy-and-paste messages and resource property elements when creating composite interfaces in WSDL 1.1. This requirement will disappear with the proposed WSDL 2.0 definition, which has the same interface extension as OGSI's GWSDL.

11 Conclusions

We have described the relationship between the concepts, mechanisms, and syntax defined by the Open Grid Services Infrastructure 1.0 specification [OGSI-Spec] and the five specifications that make up the proposed WS-Resource Framework [WS-Resource Framework] as well as the related WS-Notification family of specifications. We have discussed the rationale behind this evolution and attempted to describe the value it provides. These specifications capture all of the functionality provided by OGSI, but do so in a way that integrates with evolving Web services standards. Specifically, the WS-Resource Framework definition relies upon the WS-Addressing specification. In addition, the WS-Resource Framework definition expresses the capabilities of the OGSI definition in a way that is more consistent, and will be more familiar to Web services developers in general. Furthermore, the changes required to port an interface from OGSI to the WS-Resource Framework are few and straightforward.

12 Acknowledgements

This white paper development is the joint work of many individuals and teams. The authors wish to acknowledge the contributions from many people, including:

Sonny Fulkerson, Carl Kesselman, Susan Malaika, Martin Nally, Jeff Nick, Chris Sharp, Tony Storey, and Jay Unger. We also acknowledge those with whom we have discussed issues addressed in this paper, including Malcolm Atkinson and Savas Parastatidis.

13 References

[OGSI-Spec]

Open Grid Services Infrastructure (OGSI) V1.0 http://forge.gridforum.org/projects/ggf-editor/document/draft-ogsi-service-1/en/1

[Physiology]

Foster, I., Kesselman, C., Nick, J., Tuecke, S., The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Globus Project, 2002. Available at http://www.globus.org/research/papers/ogsa.pdf

[State Paper]

Modeling Stateful Resources using Web Services. http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf

[SOAP]

The fundamental message enveloping mechanism in Web services. http://www.w3.org/TR/SOAP**.**

[WSDM]

OASIS WSDM Management Using Web Services and Management Of Web Services Working Groups.

[WS-Addressing]

WS-Addressing, an XML serialization and standard SOAP binding for representing network wide "pointers" to services. http://www.ibm.com/developerworks/webservices/library/ws-add/

[WS-BaseFaults]

This specification defines a base fault type for use when returning faults in a Web services message exchange. It can be used when reporting faults relating to WS-Resource definition and use. This specification is a work in progress.

[WS-Arch]

The W3C Web Services Architecture working group, public draft, August 2003. http://www.w3.org/TR/2003/WD-ws-arch-20030808/

[WS-MetaDataExchange]

WS-MetadataExchange is a set of Web service mechanisms to exchange policies, WSDL, schema and other metadata between two or more parties. This specification is part of the Web services roadmap for WS-Federation. This specification is a work in progress.

[WS-Notification]

This whitepaper describes the concepts, patterns and terminology used in the WS-Notification family of specifications (WS-BaseNotification, WS-Topics and WS-BrokeredNotification).

http://www-106.ibm.com/developerworks/library/ws-pubsub/WS-PubSub.pdf

[WS-RenewableReferences]

This specification describes how a WS-Addressing endpoint reference can be decorated with information on how a new version of an endpoint reference can be retrieved by a requestor when an endpoint reference becomes invalid. This specification is a work in progress.

[WS-ResourceProperties]

This specification describes how elements of publicly visible properties of a resource can be described, retrieved, changed and deleted. http://www-106.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf

[WS-ResourceLifetime]

This specification describes a collection of message exchanges that allow a requestor to destroy a resource either immediately or by using a scheduled expiration mechanism. http://www-106.ibm.com/developerworks/library/ws-resource/ws-resourcelifetime.pdf

[WS-Security]

The roadmap to the various security related Web services standards. See: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[WS-ServiceGroup]

This specification describes a means of representing and managing heterogeneous by-reference collections of Web services or WS-Resources. This specification is a work in progress.

[WSDL]

The Web Services Description Language, version 1.1. W3C Note. See http://www.w3.org/TR/wsdl.

[WSDL 2.0]

The Web Services Description Language, version 2.0 draft. See http://www.w3.org/TR/wsdl.