

# Modeling and Managing State in Distributed Systems: The Role of OGSi and WSRF

IAN FOSTER, KARL CZAJKOWSKI, DONALD F. FERGUSON, JEFFREY FREY,  
STEVE GRAHAM, TOM MAGUIRE, DAVID SNELLING, AND STEVEN TUECKE

## Invited Paper

We often encounter in distributed systems the need to model, access, and manage state. This state may be, for example, data in a purchase order, service level agreements representing resource availability, or the current load on a computer. We introduce two closely related approaches to modeling and manipulating state within a Web services (WS) framework: the Open Grid Services Infrastructure (OGSI) and WS-Resource Framework (WSRF). Both approaches define conventions on the use of the Web service definition language schema that enable the modeling and management of state. OGSi introduces the idea of a stateful Web service 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. WSRF refactors and evolves OGSi to exploit new Web services standards, specifically WS-addressing, and to respond to early implementation and application experiences. WSRF retains essentially all of the functional capabilities present in OGSi, while changing some syntax (e.g., to exploit WS-addressing) and also adopting a different terminology in its presentation. In addition, WSRF partitions OGSi functionality into five distinct composable specifications. We explain the relationship between OGSi and WSRF and the related

WS-notification specifications, explain the common requirements that both address, and compare and contrast the approaches taken to the realization of those requirements.

**Keywords**—Grid, Open Grid Services Infrastructure (OGSI), service-oriented architecture, Web Service Resource Framework (WSRF), Web services (WS).

## I. INTRODUCTION

While Web service *implementations* are typically stateless, their *interfaces* frequently provide a user with the ability to access and manipulate state, i.e., data values that persist across, and evolve as a result of, Web service interactions. In other words, the message exchanges that Web services implement are frequently intended to enable access to stateful resources.

Web services successfully implement applications that manage state today. The messages that the services send and receive *imply* (or programmers *infer*) the existence of an associated resource type. It is desirable to define Web service conventions to enable the discovery of, introspection on, and interaction with stateful resources in standard and interoperable ways. Most important, such an approach improves the robustness of design time selection of services during application assembly and runtime binding to specific resource instances.

Open Grid Services Infrastructure (OGSI) specification version 1.0 [14], released in July 2003, defines a set of conventions and extensions for the use of Web service definition language (WSDL) [5] and XML schema [7] 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* [9], a Web service that conforms to a set of conventions for such purposes as service

Manuscript received March 1, 2004; revised June 1, 2004. This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy under Contract W-31-109-Eng-38 and in part by IBM.

I. Foster is with the Mathematics and Computer Science Division, Office of Science, Argonne National Laboratory, Argonne, IL 60439 USA and also with the Department of Computer Science, University of Chicago, Chicago, IL USA.

K. Czajkowski is with the Information Sciences Institute of the University of Southern California, Marina del Rey, CA 90292 USA (e-mail: karlcz@isi.edu).

D. F. Ferguson and S. Graham are with IBM Corporation, Research Triangle Park, NC 27713 USA.

J. Frey and T. Maguire are with IBM Corporation, Poughkeepsie, NY 12601 USA.

D. Snelling is with Fujitsu Corporation, Hayes, Middlesex UB4 8FE, U.K. (e-mail: d.snelling@fle.fujitsu.com).

S. Tuecke was with the Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439 USA. He is now with Univa, Inc., Elmhurst, IL 60126 USA.

Digital Object Identifier 10.1109/JPROC.2004.842766

**Table 1**  
Refactoring of OGSi Yields Five Normative WSRF Specifications  
Plus Three WS-Notification Specifications

Name	Description
WS-Resource Properties	Describe associating stateful resources using Web services, and how elements of publicly visible properties of a resource are retrieved, changed, and deleted.
WS-Resource Lifetime	Allows a requestor to destroy a WS-resource either immediately or at a scheduled future point in time.
WS-Renewable References	Annotate a WS-addressing endpoint reference with information needed to retrieve a new endpoint reference when the current reference becomes invalid.
WS-Service Group	Creates and uses heterogeneous by-reference collections of Web services.
WS-BaseFault	Describes a base fault type used for reporting errors.
WS-BaseNotification	Standard approaches to point-to-point notification using a publish and subscribe pattern.
WS-Topics	A standard language for describing topics in support of a publish and subscribe pattern.
WS-BrokeredNotification	Standard supporting notification intermediaries.

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 [1] progressing and the release of new draft specifications such as WS-addressing [4]. 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 [3]. Furthermore, OGSi 1.0 combined into one specification functionality that is independently useful, for example event notification. OGSi interfaces can be factored to produce a framework of independently useful Web service standards.

A recent effort aimed at achieving such a refactoring has produced the specifications listed in Table 1, five of which are named collectively the *WS-Resource Framework* (WSRF) [8], while the WS-notification family of specifications addresses 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 adopt 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.

In this paper, we explain how the new specifications derive from and relate to OGSi. To this end, we explain how each OGSi construct is realized in the new specifications and point out areas in which the new specifications provide different or enhanced functionality. In thus defining a mapping from OGSi to WSRF, we hope to persuade the OGSi community that the new specifications are a useful refactoring and evolution of OGSi; 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 identify issues that arise in migrating OGSi-based applications to WSRF and WS-notification.

In the sections that follow, we first review OGSi and related Web services specifications (Section II) and introduce the principal concepts that underlie WSRF, including the WS-resource construct (Section III). Then, we compare and contrast in turn the OGSi and WSRF treatments of service addressing (Section IV), resource properties (Section V), lifetime management (Section VI), service groups (Section VII), and faults (Section VIII), and the WS-notification treatment of notification (Section IX). Finally, we discuss issues that arise when migrating applications from OGSi to WSRF and WS-notification (Section X) and conclude. We have tried to make this paper accessible to the reader unfamiliar with OGSi and WSRF, but the reader who wishes to understand technical details will need to read the relevant technical specifications.

## II. BACKGROUND

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

### A. OGSi

OGSi is concerned primarily with creating, addressing, inspecting, and managing the lifetime of stateful *grid services* [9]. The OGSi version 1.0 specification [14], 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 definitions 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 augment the constraint capabilities of XML schema definitions. More specifically, the OGSi specification defines the following:

- 1) Set of WSDL extensions, some of which have analogous support in WSDL 2.0 [1].
- 2) WSDL constructs and standard operations for representing, querying, and updating *service data* (metadata and state data) associated with a service.

- 3) Grid service handle and grid service reference constructs, used to address grid services.
- 4) 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.
- 5) Set of operations for creating and destroying grid services that provides for both explicit destruction of services and implicit garbage collection of expired services via a “lease” or “soft-state” mechanism [10], [13] without the need for explicit destruction.
- 6) Set of operations for creating and using heterogeneous by-reference collections of Web services.
- 7) 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 early experience has been gained with the use of OGSi constructs in applications [12]. In addition, efforts have been made to develop higher level specifications that build on OGSi constructs, such as WS-agreement [6] and data access and integration services [2], [11].

### B. 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 [4] provides transport-neutral mechanisms to address Web services. Specifically, the 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 not only provides the address of the Web service itself but can also serve to identify state instances “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 by OGSi, it is beneficial to exploit those Web services specifications rather than maintaining a specification that defines the same functionality redundantly.

### C. Web Services Critiques of OGSi

While the motivation for WSRF is primarily the desire to integrate recent developments in Web services architecture, in particular, the WS-addressing design also addresses four

criticisms of OGSi from the Web services community, as we now explain.

*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. WSRF and WS-notification address this critique by partitioning OGSi v1.0 functionality into a family of specifications that allow for flexible composition.

*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. WSRF 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, WSRF 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.

*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” [15]. In addition, some Web services implementations do not accommodate dynamic service creation and destruction. WSRF rearticulates the underlying OGSi architecture to make an explicit distinction between the “service” and the stateful entities acted upon by that service. WSRF calls these entities “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.

*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 [1]. 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.

## III. FROM OGSi TO THE WS-RESOURCE FRAMEWORK

We introduce the general approach and motivations for the factoring and evolution process that takes OGSi to WSRF. This development involves three evolutionary steps as follows:

- 1) introduction of the WS-resource concept;
- 2) better separation of function and exploitation of other Web services specifications;

**Table 2**  
How Primary OGSi Constructs Map to WSRF and  
WS-Notification Constructs

OGSI	WSRF	Comments
Grid Service Reference	WS-Addressing Endpoint Reference	Uses the endpoint reference properties of WS-addressing to identify a WS-resource associated with the Web service at the designated endpoint.
Grid Service Handle	WS-Addressing Endpoint Reference & WS-Renewable References	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.
Handle Resolver portType	WS-Renewable References	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-Resource Properties	Multiple operations instead of one extensible operation, supporting simpler bindings to programming models.
GridService portType lifetime mgmt	WS-Resource Lifetime	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.
Service Group portTypes	WS-Service Group	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.

- 3) broader view of notification, which is a general Web service requirement upon which state change notification can be built.

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

#### A. WS-Resource Construct

WSRF 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 [4]. A stateful resource that participates in the implied resource pattern is termed a *WS-resource*. The framework describes the WS-resource definition and association with the description of a Web service interface and describes how to make the properties of a WS-resource accessible through a Web service interface and to manage a WS-resource’s lifetime.

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

The principal conceptual difference between the two approaches is that WSRF uses different constructs to model a stateful resource and a Web service, while OGSi uses the same construct for both by modeling stateful resources as Web services that support the GridService portType. Any pattern expressible in OGSi is expressible within WSRF: we simply arrange for a one-to-one mapping between a Web service and a single associated WS-resource.

WSRF has two advantages relative to OGSi. First, it better exploits existing XML standards, as well as emerging Web services standards such as WS-addressing. This makes WSRF easier to implement within existing and emerging Web services toolkits and easier to exploit within the myriad Web services interfaces in definition. The second advantage is pedagogical. OGSi terminology and constructs have caused anxiety for some in the Web services community because of a mistaken view that OGSi implies that Web services must become heavyweight constructs. WSRF 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 WSRF model prevents a Web services implementation from being a stateless message processor that dispatches operations to backend resources. The WSRF model makes this fact clearer, because of its more direct translation to an implementation approach that separates message processors from resources.

#### B. Other Changes

WSRF 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, WSRF 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 event system. Thus, WSRF defines the more general WS-notification specification.

OGSi uses the grid service reference as an address for a grid service and introduces the OGSi grid service handle 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. WSRF defines a framework for these functions and provides the independent mechanisms. The ability to make endpoint references stable references by the addition of endpoint policy assertions is defined in WS-RenewableReferences.

The broad scope of the OGSi specification has made it hard for readers to understand its contents and to identify the components required for a specific task. Thus, WSRF partitions OGSi functionality into distinct specifications.

OGSi uses XML Schema aggressively and, in particular, makes substantial use of extensibility (e.g., `xsd:any`). Unfortunately, this use of standard XML schema features has caused problems with some existing Web services toolkits, XML development tools, and standards (e.g., JAX-RPC). Thus, WSRF 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, WSRF 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.

#### IV. STATEFUL RESOURCE ADDRESSING

We now proceed to discuss the WSRF rendering of each OGSi construct in turn, first presenting requirements and then comparing and contrasting the OGSi and WSRF approaches to meeting those requirements. We start with addressing.

Because the WS-resources to which we wish to 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 stateful instances 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 represen-

tation 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 as a result of 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, WSRF 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 WS-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 naming of OGSi services and WS-resources provided by OGSi and WSRF, respectively, is equivalent. There may be multiple OGSi GSHs to the same service, while in WSRF there may be multiple endpoint references to the same resource. Two GSHs can be compared for equality only via syntactic comparison, but service inequality cannot

be deduced from GSH syntactic inequality. The same is true of endpoint references. Finally, nonreuse 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.

The OGSIs GSHs URI syntax provides a short human-readable “name” for a service. There is not an equivalent feature in WSRF. Instead, various forms of naming services can be built on top of WSRF. These services can provide whatever form of name is desired and map to endpoint references.

Thus, WSRF provides virtually all functionality present in OGSIs and has the advantages of leveraging WS-addressing, allowing for arbitrary hierarchies of resolver services and making clear that different features can be used independently.

## V. 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;
- 2) issue read, modify, and query requests against state components.

OGSI and WSRF 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.

WSRF 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 an interface by using an XML attribute on the WSDL 1.1 portType. In this way, the existence and type of a resource properties document are captured, as well as its association with a particular portType. This construction is legal WSDL 1.1, thanks to the revised WSDL 1.1 schema required by WS-interopability 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 OGSIs service data element declarations use an OGSIs-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 OGSIs.

**Table 3**  
Mapping From OGSIs to WSRF Lifetime Management Constructs

Function	OGSI	WSRF
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 WSRF.
Scheduled destruction	GridService portType ops, “requestTermination After” and “requestTermination Before”	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”

These attributes, if deemed critical for some applications, could be defined in some other manner such as metadata attachments, but WSRF 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. WSRF 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 WSRF for accessing resource property elements is a superset of that provided by OGSIs.

## VI. 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 OGSIs or WSRF. However, we do need to address three aspects of the entity lifecycle: creation, identity assignment, and destruction. Both OGSIs and WSRF address these three issues in essentially the same way. The mapping from OGSIs to WSRF 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 are implementation specific.

For these reasons, WSRF 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. The WSRF 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 construct. Thus, the creation of a stateful Web service in OGSI terms is really the creation of a WS-resource in WSRF terms.

A requestor that requests a factory to create a new stateful entity will typically be interested in that new entity only 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 WSRF 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 Web service identified by the endpoint reference. The WS-resource reference properties within the endpoint reference allow the Web service to identify the WS-resource targeted for destruction. 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 indicates only that destruction has been initiated.

The WSRF-defined SetTerminationTime operation supports scheduled destruction in the same way as the OGSI defined "requestTerminationAfter"; no equivalent to "request-TerminationBefore" 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 WSRF address these requirements in the same manner: via two service data elements (OGSI) or resource properties (WSRF), CurrentTime and TerminationTime.

## VII. SERVICE GROUPS

The term *service group* refers to a standard mechanism for creating a heterogeneous by-reference collection of services

or resources. Service groups can form a wide variety of collections of services or resources, including building registries of services and resources. While their use is not restricted to grid services or WS-resources, service groups are particularly important when dealing with stateful entities.

OGSI and WSRF 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.

## VIII. FAULTS

WSDL defines a message exchange fault model, but not a base format for fault messages. Specific domains that define interfaces using WSDL typically define a "fault schema" for utilization across various message exchanges defined in those interfaces. Both OGSI and WSRF 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.

WSRF 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 on the capabilities of broadly available Web services tooling.

## IX. NOTIFICATION

In an environment in which stateful resources may be created and destroyed, or may change 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.

Notification is a broad concept. Not all "events" relate to changes in the "state" of a service or resource. WS-notification introduces a more feature-complete, generic, topic hierarchy approach for publish/subscribe-based notification, which is a common model followed in large-scale distributed

event management systems. WS-resource properties then define 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.

## X. PORTING INTERFACES FROM OGSi TO WSRF

We have, as yet, little experience in porting interfaces from OGSi to WSRF and WS-notification. However, we expect that this process will be, in general, straightforward, comprising a set of mechanical transformations.

WSDL operation definitions themselves need not change. What changes is 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.

## XI. CONCLUSION

We have described the relationship between the concepts, mechanisms, and syntax defined by the OGSi 1.0 specification and the five specifications that make up the proposed WSRF as well as the related WS-notification family of specifications. We have discussed the rationale behind the evolution from OGSi to WSRF and outlined the value that WSRF provides.

As we have described, WSRF and WS-notification capture all of the functionality provided by OGSi, but do so in a way that integrates better with evolving Web services standards. Specifically, the WSRF definition relies upon the WS-addressing specification. In addition, the WSRF 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.

## ACKNOWLEDGMENT

The authors would like to thank S. Fulkerson, C. Kesselman, S. Malaika, M. Nally, J. Nick, C. Sharp, T. Storey, and J. Unger for their contributions. They also acknowledge those with whom they have discussed issues addressed in this paper, including M. Atkinson and S. Parastatidis.

## REFERENCES

- [1] The Web Services Description Language (2004). [Online]. Available: <http://www.w3.org/TR/wsdl>
- [2] M. Atkinson, A. Chervenak, P. Kunszt, I. Narang, N. Paton, D. Pearson, A. Shoshani, and P. Watson, "Data access, integration, and management," in *The Grid: Blueprint for a New Computing Infrastructure*. San Diego, CA, 2004.
- [3] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. (2003) Web Services Architecture. [Online]. Available: <http://www.w3.org/TR/2003/WD-ws-arch-20030808/>

- [4] A. Bosworth, D. Box, E. Christensen, F. Curbera, D. Ferguson, J. Frey, C. Kaler, D. Langworthy, F. Leymann, S. Lucco, S. Millet, N. Mukhi, M. Nottingham, D. Orchard, J. Shewchuk, E. Sindambiwe, T. Storey, and S. Weerawarana. (2004) Web Services Addressing (WS-Addressing). [Online]. Available: <http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810>
- [5] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. (2001) Web Services Description Language (WSDL). [Online]. Available: [www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl)
- [6] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu, Agreement-Based Grid Service Management (WS-Agreement), Global Grid Forum, Draft, 2003.
- [7] D. C. Fallside. (2001) XML Schema Part 0: Primer. [Online]. Available: [www.w3.org/TR/xmlschema-0](http://www.w3.org/TR/xmlschema-0)
- [8] I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. Ferguson, F. Leymann, M. Nally, T. Storey, and S. Weerawarana, *Modeling Stateful Resources with Web Services*: Globus Alliance, 2004.
- [9] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "Grid services for distributed systems integration," *IEEE Comput.*, vol. 35, no. 6, pp. 37–46, Jun. 2002.
- [10] C. G. Gray and D. R. Cheriton, "Lease: An efficient fault-tolerant mechanism for distributed file cache consistency," in *Proc. 12th ACM Symp. Operating System Principles*, 1989, pp. 202–210.
- [11] N. W. Paton, M. P. Atkinson, V. Dialani, D. Pearson, T. Storey, and P. Watson. (2002) *Database Access and Integration Services on the Grid* [Online]. Available: [www.nesc.ac.uk](http://www.nesc.ac.uk)
- [12] L. Pearlman, C. Kesselman, S. Gullapalli, B. F. Spencer, J. Futrelle, K. Ricker, I. Foster, P. Hubbard, and C. Severance, "Distributed hybrid earthquake engineering experiments: Experiences with a ground-shaking grid application," in *Proc. 13th IEEE Int. Symp. High Performance Distributed Computing*, 2004.
- [13] S. Raman and S. McCanne, "A model, analysis, and protocol framework for soft state-based communication," in *ACM SIGCOMM Computer Communication Rev.*, vol. 29, 1999, pp. 15–25.
- [14] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, D. Snelling, and P. Vanderbilt. (2003) Open Grid Services Infrastructure (OGSi) Version 1.0.. Global Grid Forum. [Online]. Available: [www.ggf.org](http://www.ggf.org)
- [15] W. Vogels, "Web services are not distributed objects: Common misconceptions about the fundamentals of Web service technology," *IEEE Internet Comput.*, vol. 7, no. 6, pp. 59–66, 2003.



**Ian Foster** is the Associate Director of the Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, and the Arthur Holly Compton Distinguished Service Professor of Computer Science at the University of Chicago, Chicago, IL. His research interests include distributed and parallel computing and computational science, and he has published six books and over 200 articles and technical reports on these and related topics. His Distributed Systems Laboratory is home to the Globus Toolkit,

widely used open source grid software, and also plays a leading role in projects applying grid technologies to scientific and engineering problems, in such fields as high energy physics, climate data analysis, and earthquake engineering.

Dr. Foster is a Fellow of the American Association for the Advancement of Science and the British Computer Society. His awards include the British Computer Society's award for technical innovation, the Global Information Infrastructure (GII) Next Generation award, the British Computer Society's Lovelace Medal, and Research and Development Magazine's Innovator of the Year.

**Karl Czajkowski** received the B.A. degree in computer science from the University of California, Berkeley, in 1996.

He joined USC/ISI as a Software Developer and was a Computer Scientist in the Center for Grid Technologies working with the Globus Project for seven years. He is now with the Univa Corporation, Elmhurst, IL, continuing his work with the Globus Alliance as a Software Architect and Globus Alliance Board member. His research interests include the current and future Globus Toolkit components for grid scheduler negotiation and service management.



**Donald F. Ferguson** received the Ph.D. degree in computer science from Columbia University, New York, in 1989, with a thesis that studied the application of economic models to the management of system resources in distributed systems.

He joined IBM Watson Research in 1987 and from 1993 has focused his efforts in the area of distributed and object-oriented systems, work that produced the IBM Component Broker and WebSphere family of products. He is one of 59 IBM Fellows in IBM's engineering community

of 160,000 technical professionals. He is the chief architect for IBM's WebSphere Platform and Chairman of IBM Software Group's Architecture Board. His most recent efforts have focused on Web services, business process management, Grid services, and application development for WebSphere. He has over 24 technical publications and seven granted or pending patents.

Dr. Ferguson was the Co-program chairman for the First International Conference on Information and Computation Economics and has received a best paper award for work on database buffer pools.



**Jeffrey Frey** received the B.S. degree in computer science from the Rochester Institute of Technology, Troy, NY.

He is a Distinguished Software Engineer at IBM, Poughkeepsie, NY, and a member of the IBM Academy of Technology. He joined IBM in 1983 as a Programmer working on the development of IBM's MVS operating system kernel. During his career at IBM he has helped to define and implement many of IBM's technical initiatives. He was one of a small number of

architects responsible for IBM's large system clustering technologies in support of high-performance data sharing in commercial systems and also served as the Chief Architect for IBM's WebSphere/390 application server product. His key assignments have been in the areas of MVS and zOS operating system development, IBM's Parallel Sysplex architecture, distributed systems infrastructure, distributed OO and component-based transactional application servers, Web services architecture, and autonomic systems management. Currently, he is a Lead Architect in IBM's On Demand and autonomic computing strategy.



**Steve Graham** received the B.Math. and M.Math. degrees in computer science from the University of Waterloo and the MBA degree from the Kenan Flagler Business School, University of North Carolina, Chapel Hill.

He was a developer with Sybase, a consultant, and a faculty member in the Department of Computer Science at the University of Waterloo; an architect and consultant with the IBM Smalltalk consulting organization; and a technologist and consultant working with various emerging

technologies such as Java and XML. He is currently a Senior Technical Staff Member in IBM's Software Group and member of the IBM Academy of Technology. He is a Web services architect working in standards and with the On Demand Architecture group. He has spent the last four years working on service-oriented architectures, as part of the IBM's Web Services Initiative and IBM's On Demand Initiative. He is the lead author of *Building Web Services With Java: Making Sense of XML, SOAP, WSDL and UDDI, 2nd Edition* (Indianapolis, IN: Sams, 2004). He can be reached at [ssgraham@us.ibm.com](mailto:ssgraham@us.ibm.com).



**Tom Maguire** is a Senior Technical Staff Member in IBM's Systems Group, Poughkeepsie, NY, and an Architect in the On Demand Architecture group. He has spent the past two years working on IBM's On Demand architecture. He is a coauthor on several emerging OASIS Web Service specifications, notably Web Services Resource Framework (WSRF) and Web Services Notification (WSN). He is an active member of OASIS, Global Grid Forum (GGF), and DMTF workgroups and technical

committees. Earlier notable work includes IBM's Globus Toolkit product offering IBM Grid Toolbox and coauthorship of the Open Grid Services Infrastructure (OGSI) specification. Tom has also worked as an Architect and Lead Designer on WebSphere Commerce catalog management tools. Before joining IBM, he was a Consultant in object-oriented design and modeling, ontology, and knowledge representation.



**David Snelling** received the B.S. and M.S. degrees in mathematics from the University of Denver, Denver, CO, in 1979 and 1981, respectively, and the Ph.D. degree in computer science, Manchester University, U.K., in 1993.

Since June 2002, he has been a Manager of the Distributed Services Research Group, Fujitsu Laboratories, U.K. From January 1997 to June 2002, he was a Research Manager of the Fujitsu European Centre for Information Technology (in June 2002, Fujitsu Laboratories of Europe subsumed FECIT). From January 1996 to December 1996, he was a Lecturer in computer science at the University of Manchester, Manchester, U.K., and prior to that, also at the University of Manchester, was a Research Fellow in computer science, from May 1992 to December 1995. He was a Lecturer in computer science at the University of Leicester, Leicester, U.K., from September 1987 to April 1992 and was granted tenure in 1989. From January 1986 to August 1987, he was a Consultant in Multiprocessing, European Centre for Medium Range Weather Forecasts, Reading, U.K. From September 1982 to December 1985, he was an Applications Analyst at Denelcor Limited, Richmansworth, U.K., and prior to that was a Software Analyst at the same company, but in Denver. From September 1979 to June 1981, he was a Graduate Teaching Assistant at the University of Denver, and from June 1975 to July 1981 was a Programmer for the National Oceanic and Atmospheric Administration, Boulder, CO. His research interests include grid computing and distributed service architectures, object-oriented systems analysis, and advanced personal information systems.

He is the Architecture Area Director and member of the Global Grid Forum Steering Group, GGF. He is co-chair of the Web Services Resource Framework Technical Committee, OASIS, co-chair of the Open Grid Service Infrastructure Working Group, GGF, and a participant on OGSA-WG (GGF), WSN-TC (OASIS), WSDM-TC (OASIS), Attributes Task Force WSDL-TC (W3C). He is also Cochair of the EU "Next Generation Grids Expert Group." He is on the OMII Technical Advisory Board, EGEE External Advisory Committee, Unicore Technical Board, and OGSA DAI Program Board and is a regular member of various program committees.



**Steven Tuecke** received the B.A. degree (*summa cum laude*) in mathematics and computer science from St. Olaf College.

He is CEO of Univa Inc., Elmhurst, IL, a company focused on providing commercial support for the open source Globus Toolkit. Prior to the founding of Univa in 2004, he was lead Software Architect in the Distributed Systems Laboratory (DSL), Mathematics and Computer Science Division, Argonne National Laboratory, working primarily on Grid research and development. He was

a Chief Architect of the Globus Toolkit and is actively involved in standards work at the Global Grid Forum and Internet Engineering Task Force.