The WS-Resource Framework

Version 1.0

03/05/2004

Authors

Karl Czajkowski (Globus Alliance / USC Information Sciences Institute)

Donald F Ferguson (IBM)

Ian Foster (Globus Alliance / Argonne National Laboratory)

Jeffrey Frey (IBM)

Steve Graham (IBM)

Igor Sedukhin (Computer Associates International)

David Snelling (Fujitsu Laboratories of Europe)

Steve Tuecke (Globus Alliance / Argonne National Laboratory)

William Vambenepe (Hewlett-Packard)

Abstract

The WS-Resource construct has been proposed as a means of expressing the relationship between stateful resources and Web services. We introduce here the WS-Resource framework, a set of proposed Web services specifications that define a rendering of the WS-Resource approach in terms of specific message exchanges and related XML definitions. These specifications allow the programmer to declare and implement the association between a Web service and one or more stateful resources. They describe the means by which a view of the state of the resource is defined and associated with a Web services description, forming the overall type definition of a WS-Resource. They also describe how the state of a WS-Resource is made accessible through a Web service interface, and define related mechanisms concerned with WS-Resource grouping and addressing. This paper provides an architectural overview of the WS-Resource framework. It motivates, introduces, and summarizes the interrelationships among five separate specification documents that provide the normative definition of the framework: WS-ResourceProperties, WS-ResourceLifetime, WS-RenewableReferences, WS-ServiceGroup, and WS-BaseFaults. We also describe how the WS-Resource framework can support the WS-Notification family of specifications for asynchronous notification.

Status

This whitepaper is an initial draft release and is provided for review and evaluation only. The authors hope to solicit your contributions and suggestions in the near future. The authors make no warrantees or representations regarding the specifications in any manner whatsoever.

Table of Contents

1 Introduction

Web service 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. However, the notion of stateful resources acted upon by the Web service implementation is not explicit in the interface definition. The messages that the services send and receive *imply* (or encourage programmers to *infer)* the existence of an associated stateful 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 importantly, such an approach improves the robustness of design-time selection of services during application assembly and runtime binding to specific stateful resource instances.

These observations motivated the proposed *WS-Resource* approach to modeling state in a Web services context [WS-Resource]. A WS-Resource is defined as the composition of a Web service and a stateful resource that is (i) expressed as an association of an XML document with defined type with a Web services portType, and (ii) addressed and accessed according to the implied resource pattern, a conventional use of WS-Addressing endpoint references. In the implied resource pattern, a stateful resource identifier is encapsulated in an endpoint reference and used to identify the stateful resource to be used in the execution of a Web service message exchange. The WS-Resource framework allows WS-Resources to be declared, created, accessed, monitored for change, and destroyed via conventional Web services mechanisms, but does not require that the Web service component of the WS-Resource that provides access to the associated stateful resources be implemented as a stateful message processor.

In this paper, we introduce the WS-Resource framework, a set of five technical specifications that define the normative description of the WS-Resource approach in terms of specific Web services message exchanges and related XML definitions. These technical specifications, summarized in Table 1, define the means by which:

- a WS-Resource can be destroyed, either synchronously with respect to a destroy request or through a mechanism offering time-based (scheduled) destruction, and specified resource properties [WS-ResourceProperties] may be used to inspect and monitor the lifetime of a WS-Resource (WS-ResourceLifetime);
- the type definition of a WS-Resource can be composed from the interface description of a Web service and an XML resource properties document, and the WS-Resource's state can be queried and modified via Web services message exchanges (WS-ResourceProperties);
- a Web service endpoint reference (WS-Addressing) can be renewed in the event the addressing or policy information contained within it becomes invalid or stale (WS-RenewableReferences);
- heterogeneous by-reference collections of Web services can be defined, whether or not the services are WS-Resources (WS-ServiceGroups); and

- fault reporting can be made more standardized through use of an XML Schema type for base faults and rules for how this base fault type is used and extended by Web services (WS-BaseFaults).

In the rest of this paper, we first introduce the WS-Resource construct and the implied resource pattern, and then summarize each of the five WS-Resource framework technical specifications in turn. We also discuss how an publish/subscribe (pub/sub) notification mechanism (WS-Notification) can be built on top of the WS-Resource framework, by creating subscriptions to state changes within the WS-Resource to monitor and report on resource property changes.

The WS-Resource framework is inspired by the work of the Global Grid Forum's Open Grid Services Infrastructure (OGSI) Working Group [Physiology, OGSI-Spec]. Indeed, it can be viewed as a straightforward refactoring of the concepts and interfaces developed in the OGSI V1.0 specification in a manner that exploits recent developments in Web services architecture (e.g., WS-Addressing). We discuss the relationship between the WS-Resource framework and OGSI elsewhere [OGSI-Refactor].

Table 1: The WS-Resource Framework is defined by five normative specifications, each described in a section of this paper.

Name	Describes	Section
WS-ResourceLifetime	Mechanisms for WS-Resource destruction, including message exchanges that allow a requestor to destroy a WS-Resource, either immediately or by using a time-based scheduled resource termination mechanism.	3
WS-ResourceProperties	Definition of a WS-Resource, and mechanisms for retrieving, changing, and deleting WS-Resource properties.	4
WS-RenewableReferences	A conventional decoration of a WS-Addressing endpoint reference with policy information needed to retrieve an updated version of an endpoint reference when it becomes invalid.	5
WS-ServiceGroup	An interface to heterogeneous by-reference collections of Web services.	6
WS-BaseFaults	A base fault XML type for use when returning faults in a Web services message exchange.	7

2 WS-Resource and Implied Resource Pattern

The WS-Resource definition codifies the relationship between Web services and stateful resources in terms of the *implied resource pattern*, a set of conventions on Web services technologies, particularly XML, WSDL, and WS-Addressing [WS-Addressing]. These conventions allow the state of a resource that participates in the implied resource pattern to be defined and associated with the description of a Web service interface. The state of a resource is defined in terms of a resource properties document. We summarize these conventions here; more details are provided elsewhere [WS-Resource]. We use the term *WS-Resource* to describe this pairing of a Web service and a resource properties document.

WS-Addressing standardizes the endpoint reference construct used to represent the address of a Web service deployed at a given network endpoint. An endpoint reference may contain, in addition to the endpoint address of the Web service, other metadata associated with the Web service such as service description information and *reference properties*, which help to further qualify the use of the Web service address. The reference properties of the endpoint reference play an important role in the implied resource pattern.

The implied resource pattern defines a conventional use of WS-Addressing in which a stateful resource is treated as an implied input for the processing of message exchanges implemented by a Web service.

An endpoint reference that is described as following the implied resource pattern may include a ReferenceProperties child element that identifies the stateful resource to be used in the execution of all message exchanges performed using this EndpointReference. This type of endpoint reference is referred to as a *WS-Resource-qualified endpoint reference*. A request message directed to a Web service designated by a WS-Resource-qualified endpoint reference must include the ReferenceProperties information from the endpoint reference, as specified by WS-Addressing.

Thus, the WS-Resource framework uses a WS-Resource-qualified endpoint reference to represent a "network-wide pointer" to a WS-Resource. A WS-Resource-qualified endpoint reference may be returned as a result of a Web service message request to a factory to *create* a new WS-Resource or, alternatively, from the evaluation of a search query on a service registry, or as a result of some application-specific Web service request.

3 WS-Resource Lifecycle and Identity

The WS-ResourceLifetime specification addresses three important aspects of the WS-Resource lifecycle, namely creation, identity, and destruction.

3.1 The WS-Resource Factory Pattern

The WS-Resource framework does not attempt to define the message exchanges used to request creation of new WS-Resources. Instead, it simply notes that new WS-Resources may be created by some out-of-band mechanism, or through the use of a use pattern for Web services that the WS-ResourceLifetime specification calls a *WS-Resource factory*. (The Factory Pattern is a commonly used creational pattern [Design Patterns].) A *WS-Resource factory* is any Web service capable of bringing one or more WS-Resources into existence. The response message of a WS-Resource factory operation typically contains at least one endpoint reference that refers to the new WS-Resource, though a factory may convey the reference to the new WS-Resource through other means such as placing the WS-Resource-qualified endpoint reference(s) into a registry for later retrieval.

Note that there may be many types of Web services (e.g., resource registries) that return WS-Resource-qualified endpoint references in their response messages. However, a message exchange is only considered a WS-Resource factory operation if it results in the actual creation of the WS-Resource referred to in the returned WS-Resource-qualified endpoint reference.

3.2 WS-Resource Identity

We describe and contrast the role and use of WS-Resource identity from the perspectives of (i) the WS-Resource implementation, and (ii) a service requestor to whom an endpoint reference to a WS-Resource is provided.

Recall that, as stated in Section **Error! Reference source not found.**, the stateful resource component of a WS-Resource is identified through the use of a stateful resource identifier carried in the reference properties component of an endpoint reference. The form and contents of the stateful resource identifier carried in the reference properties is completely encapsulated within the WS-Resource implementation.

The Web service component of a WS-Resource can construct an address for the WS-Resource by including an identifier of the stateful resource in the reference properties component of a WS-Addressing endpoint reference. That endpoint reference is then said to be *WS-Resource-qualified*. The WS-Resource-qualified endpoint reference can then be made available to other entities in a distributed system, which can subsequently use that endpoint reference to direct requests to the WS-Resource. Logically, these requests "flow" through the Web service component of the WS-Resource, which understands the content of the implementation-dependent stateful resource identifier encapsulated in the WS-Address endpoint reference. Part of the WS-Address endpoint reference identifies the service, which in turn uses the reference properties to identify the stateful resource to be used in message execution.

In contrast, a service requestor that obtains access to a WS-Resource-qualified endpoint reference should not examine or attempt to interpret the contents of the reference properties that represent the stateful resource identifier. Even an attempt by the service requestor to compare two stateful resource identifiers is considered invalid. From the perspective of the service requestor, the content of the reference properties component of a WS-Resource's endpoint reference is opaque.

So how would a service requestor reason about the identity of a stateful resource component of a WS-Resource? The short answer is that the semantic meaning of the identity of the stateful resource component of a WS-Resource, and the means by which it is defined and exposed to a service requestor, is WS-Resource implementation dependent. At the current time, there are no adopted Web service specifications that provide for the definition of stateful resource identity. Nor is there any definition of the means by which the identity of a stateful resource is obtained by a service requestor.

Whether or not the identity of a stateful resource component of a WS-Resource is exposed to a service requestor is WS-Resource design dependent. However, we believe many WS-Resources will provide the ability to retrieve the identity. The identity should be a portable, namespace-scoped value. Portability is important as it allows one application to pass the identity to another. Namespace scoping is important as it allows for disambiguation of multiple identities that may originate from different sources.

A Web service requestor that receives a WS-Addressing endpoint reference may pass that endpoint reference to other services, with the assurance that the receiver may invoke operations involving the WS-Resource *instance.* This is fundamental to WS-Addressing.

We envision that a common approach for exposing the identity of the stateful resource component of a WS-Resource will be to treat the identity as one or more resource properties expressed in the WS-Resource properties document. This approach would allow a service requestor to direct a query against the document, targeting the properties understood to represent the identity of the stateful resource. If the identity is exposed as one or more resource properties, the WS-Resource should ensure read-only access to those properties. Typically, it would be invalid to allow a service requestor to change the identity of a stateful resource.

As another option, the WS-Resource may implement application-specific message exchanges intended to provide access to the identity of the stateful resource component. We anticipate that many applications will recognize the need to introduce message exchanges related to stateful resource identity. Some such exchanges may provide for retrieving identity, and some may provide comparison and equality checks.

3.3 WS-Resource Destruction

A requestor that asks a WS-Resource factory to create a new WS-Resource will typically only be interested in that new WS-Resource for some finite period. After that time, it should be possible to destroy the WS-Resource so that its associated system or application resources can be reclaimed. The WS-Resource framework standardizes two lifetime management approaches for the destruction of a WS-Resource: *immediate* and *scheduled* destruction.

3.3.1 Immediate Destruction

In many scenarios, it is appropriate for applications making use of a WS-Resource to request that it be destroyed immediately. A service requestor that wishes to destroy a WS-Resource immediately must use the appropriate WS-Resource-qualified endpoint reference to send a destroy request message to the WS-Resource identified by the endpoint reference. The stateful resource identifier within the endpoint reference is used to identify the stateful resource component to be destroyed. Note that the destruction of the stateful resource component of a WS-Resource effectively destroys the WS-Resource. The immediate destroy message exchange is defined in the WS-ResourceLifetime specification [WS-ResourceLifetime].

The receipt of the response to the destroy request message represents a point of synchronism between the service requestor and the WS-Resource receiving the destroy request message. Upon receipt of the response message, any further attempted message exchanges with the WS-Resource must result in an *Unknown Resource* fault message, absent any other fault conditions that may take precedence.

3.3.2 Scheduled Destruction

A requestor may be unwilling to destroy a WS-Resource immediately and synchronously, or indeed may be unable to do so in a distributed computing environment, because the requestor has become disconnected from the service provider’s endpoint. Thus, in addition to the ability to destroy a WS-Resource immediately, we define the means by which a WS-Resource may be scheduled for termination at a future time. Using a WS-Resource-qualified endpoint reference, a service requestor may first establish and subsequently renew the scheduled termination time of the WS-Resource. When that time expires, the WS-Resource may *self destruct* without the need for a synchronous destroy request from a service requestor. A requestor may periodically update the scheduled termination time (subject to policy) to adjust the lifetime of the WS-Resource. The WS-ResourceLifetime specification [WS-ResourceLifetime] defines a standard message exchange by which a service requestor can initially establish and subsequently change the scheduled termination time of a WS-Resource, and defines a means to determine the current scheduled termination time of a WS-Resource.

A WS-Resource factory may support the ability to negotiate an initial WS-Resource scheduled termination time when a WS-Resource is created. Subsequently, authorized service requestors may use the WS-Resource-qualified endpoint reference to request that this scheduled termination time be changed, by sending the request message to the WS-Resource. If the WS-Resource’s termination time is reached, it may be destroyed and any associated system resources may be reclaimed.

A WS-Resource’s termination time may change non-monotonically. That is, a service requestor may request a termination time that is earlier than termination time already associated with the WS-Resource. If the requested termination time represents a time prior to the current time, as known by the WS-Resource implementation, the request must be interpreted as a request for immediate, but not synchronous, destruction of the WS-Resource.

4 WS-Resource Properties

The WS-ResourceProperties specification defines the type and values of those components of a WS-Resource’s state that can be viewed and modified by service requestors through a Web services interface. The key ideas are as follows.

- The WS-Resource has an XML *resource property document* defined using XML schema.
- Service requestors may determine a WS-Resource’s type by retrieving the WSDL portType definition by standard means.
- Service requestors may use Web services message exchanges to read, modify, and query the XML document representing the WS-Resource’s state.

We use the term *resource property* to refer to an individual component of a WS-Resource’s state. We call the XML document describing the type of a stateful resource component of a WS-Resource a *WS-Resource properties* document. Each resource property is represented as an XML element within the WS-Resource properties document.

The use of XML is *logical.* The underlying state may be in any or multiple formats, and in a single or multiple locations.

4.1 WS-Resource Properties Document

The WS-Resource properties document acts as a view on, or projection of, the actual state of the WS-Resource. The document serves to define the structure upon which service-requestor-initiated query and update messages can be directed. Any operation that manipulates a resource property via the WS-Resource properties document must be reflected in the actual implementation of the WS-Resource's state.

The WS-Resource properties document is described using XML Schema. Specifically, the WS-Resource properties document is expressed as an XML global element declaration (GED) in some XML namespace, comprising a set of references to XML GEDs of the individual resource properties. For example, consider a stateful resource named "C." If the state of "C" comprises three resource property components, named p1, p2, and p3, then its resource properties document, named "ExampleResourceProperties," might be defined as follows.

```
<xs:schema
  targetNamespace="http://example.com/ResourcePropertiesExample"
  xmlns:tns="http://example.com/ResourcePropertiesExample"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
…
... >

  <xs:element name="p1" type= … />
  <xs:element name="p2" type= …/>
  <xs:element name="p3" type= … />

  <xs:element name="ExampleResourceProperties">
    <xs:complexType>
       <xs:sequence>
         <xs:element ref="tns:p1" />
         <xs:element ref="tns:p2" />
         <xs:element ref="tns:p3" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
…
</xs:schema>
```

Service requestors may obtain and examine this XML schema definition of the WS-Resource properties document, which represents the type of stateful resource "C," by various means, including message exchanges defined by WS-MetaDataExchange [WS-MetaDataExchange].

But how did the service requestor know that the GED named "ExampleResourceProperties" defines the resource properties document of the WS-Resource? The resource properties document declaration for the WS-Resource occurs in the WSDL portType definition of the Web service component of the WS-Resource. The WS-Resource properties document declaration is associated with the WSDL portType definition via the use of the ResourceProperties attribute, as in the following example.

```
<wsdl:definitions
  targetNamespace="http://example.com/ResourcePropertiesExample"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wsrp=
   "http://www.ibm.com/xmlns/stdwip/web-services/ws-resourceProperties"
  xmlns:tns="http://example.com/ResourcePropertiesExample"
…>
…
  <wsdl:types>
    <xs:schema>
      <xs:import
        namespace="http://example.com/ResourcePropertiesExample"
        schemaLocation="…"/>
   </xs:schema>
  </wsdl:types>
…

  <wsdl:portType name="SomePortTypeName"
    wsrp:ResourceProperties="tns:ExampleResourceProperties" >
    <operation name="…
…
  </wsdl:portType>
…
</wsdl:definitions>
```

This association between the portType and resource properties document effectively defines the type of the WS-Resource.

4.2 WS-Resource Property Composition

The Web service interfaces associated with the various specifications related to WS-Resource have been designed to be composable. In WSDL 1.1, the designer of a Web service interface composes the interface by a copy-and-paste of the operations defined in the constituent portTypes used in the composition. For example, the operations defined in an example portType "foo" can be combined with the operations defined in various standards and specifications to yield a final, complete set of message exchanges to be implemented by a Web service.

In addition to operation composition, the designer may also aggregate the WS-Resource properties defined in the WS-Resource properties documents of the various constituent portTypes to yield the final, complete WS-Resource property document declared with the final composed portType. The designer may use any approach to XML document composition to define the final document. Examples include extension and aggregation.

This WS-Resource properties document composition may be accomplished by adding (aggregating) additional XML element declarations, using the xs:ref attribute, as demonstrated in the following example.

```
  <xs:element name="ExampleResourceProperties">
    <xs:complexType>
       <xs:sequence>
         <xs:element ref="tns:p1" />
         <xs:element ref="tns:p2" />
         <xs:element ref="tns:p3" />

         <xs:element ref="xxxx:SomeAdditionalResourceProperty"
             xmlns:xxxx= … />
```

```
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

This WS-Resource properties document was constructed by combining the resource property elements of the WS-Resource properties document for stateful resource “C” with a resource property element (SomeAdditionalResourceProperty) defined in some other namespace.

4.3 Accessing WS-Resource Property Values

The state of a WS-Resource, i.e., the values of resource properties exposed in the WS-Resource’s resource properties document, can be read, modified, and queried by using standard Web services messages. These message exchanges are defined in the WS-ResourceProperties specification and should be included as WSDL operations in any portType that uses the wsrp:ResourceProperties attribute to declare a WS-Resource properties document. We describe these operations briefly here; the WS-ResourceProperties specification provides more details [WS-ResourceProperties].

The base functionality is to retrieve the value of a single resource property using a simple Web services request/response message exchange, identifying the WS-Resource using a WS-Resource-qualified endpoint reference as described previously and identifying the resource property by qualified name of its GED. A slightly more sophisticated variant of this retrieval function allows the retrieval of the value of multiple resource properties with a single request/response message exchange.

These functions are encoded in the WS-ResourceProperties operation “get,” which allows the requestor to retrieve the value of one or more WS-Resource properties as specified in the get request message, as follows.

```
  <wsrp:GetMultipleResourceProperties>
    <wsrp:ResourceProperty>QName <wsrp:ResourceProperty>+
  </wsrp:GetMultipleResourceProperties>
```

The response to this message is a sequence of XML elements, corresponding to the values of the WS-Resource properties identified by the QNames specified in the request message.

For example, the following message represents a request to retrieve the value of the property “p1” from a stateful resource such as “C.”

```
<soap:Envelope>
  <soap:Header>
    <tns:resourceID> C </tns:resourceID>
  </soap:Header>
  <soap:Body>
    <wsrp:GetMultipleResourceProperty>
      <wsrp:ResourceProperty>tns:p1</wsrp:ResourceProperty>
    </wsrp:GetMultipleResourceProperty>
  </soap:Body>
</soap:Envelope>
```

Remember that the endpoint reference used to designate the target of this request message contains an identifier in the ReferenceProperties component that identifies the stateful resource named “C.” This identifier information is carried in the SOAP header element in the example, which is the standard encoding for WS-Addressing endpoint reference ReferenceProperties.

The WS-Resource would respond with a message containing the values of the WS-Resource property named p1 as follows.

```
<soap:Envelope>
  <soap:Body>
    <wsrp:GetMultipleResourcePropertyResponse>
      <p1>xyz</p1>
    </wsrp:GetMultipleResourcePropertyResponse>
  </soap:Body>
</soap:Envelope>
```

The WS-ResourceProperties specification also defines a "set" operation that allows the values of WS-Resource properties to be inserted, updated, and deleted. The following is a brief pseudo-syntax for the "set" request message.

```
  <wsrp:SetResourceProperties>
   {
    <wsrp:Insert >
      {any}*
    </wsrp:Insert> |

    <wsrp:Update >
      {any}*
    </wsrp:Update> |

    <wsrp:Delete ResourceProperty=”QName” />
   }+
  </wsrp:SetResourceProperties>
```

A fourth operation defined in the WS-ResourceProperties specification allows the service requestor to execute an arbitrary query expression (such as an XPath 1.0 query expression) against a WS-Resource property document. Various query expression types may be used, for example, to support resource discovery based on the current values of a WS-Resource's state.

Finally, the WS-ResourceProperties specification also defines the means by which a service requestor can subscribe for notifications of value changes in the resource properties of a WS-Resource. This is done by defining a particular use of the capabilities defined in the WS-Notification family of specifications [WS-Notification].

5 Renewable References

The WS-RenewableReferences work must define mechanisms that can be used to renew an endpoint reference that has become invalid. These mechanisms can be applied to any endpoint reference, but are particularly useful in the case of an endpoint reference that refers to a WS-Resource, as it can provide a persistent and stable reference to the WS-Resource that can allow the same state to be accessed repeatedly over time.

A WS-Addressing endpoint reference may contain not only addressing but also policy information concerning interactions with the service. Typically, endpoint references are constructed by an authoritative source of the addressing and policy information. An endpoint reference made available to a client represents a copy of that 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. In such situations, it becomes important to be able to *renew* the endpoint reference.

WS-RenewableReferences should define a specific WS-Policy assertion for the purpose of decorating endpoint references with information necessary to retrieve a new endpoint reference in the event the reference becomes invalid.

6 Service Groups

The WS-ServiceGroup specification must define a means of representing and managing heterogeneous by-reference collections of Web services. This specification can be used to organize collections of WS-Resources, for example to build registries, or to build services that can perform collective operations on a collection of WS-Resources.

The ServiceGroup specification can express ServiceGroup membership rules, membership constraints, and classifications using the resource property model from WS-ResourceProperties. Groups can be defined as a collection of members that meet the constraints of the group as expressed through resource properties. The ServiceGroup specification should also define interfaces for managing the membership of a ServiceGroup.

The interfaces defined by WS-ServiceGroup are expected to be composed with other Web services interfaces, which define more specialized interaction with the service group and/or with the services that are members of the ServiceGroup. For example, specialized interfaces may other means of querying the contents of the ServiceGroup, and for performing collective operations across members of the ServiceGroup.

7 Base Faults

The WS-BaseFaults specification must define a base fault type for use when returning faults in a Web services message exchange. While there is nothing specific to WS-Resources in this specification, it is nonetheless used by all of the other WS-Resource framework specifications to bring consistency to the faults returned by the operations in these specifications, including consistent reporting of faults relating to WS-Resource definition and use.

8 Notification

A separate family of specifications, called WS-Notification, defines a general, topic-based Web service system for publish and subscribe (pub/sub) interactions that builds on the WS-Resource framework. The basic approach taken is to define mechanisms and interfaces that allow clients to subscribe to topics of interest, such as resource property value changes for a WS-Resource. From the perspective of WS-Notification, the WS-Resource framework thus provides useful building blocks for representing and structuring notifications. From the perspective of the WS-Resource framework, the WS-Notification family of specifications extends the utility of WS-Resources by allowing requestors to ask to be asynchronously notified of changes to resource property values.

One WS-Notification specification, WS-BaseNotification, describes the basic roles, concepts, and patterns required to allow a *subscriber* to register interest in receiving notification messages from a notification *producer*. An notification can concern anything, a change in the value of a resource property, some other internal change in the state of the notification producer, or some other "situation" within the environment. A subscriber registers interest in receiving notification messages on one or more topics by issuing a "subscribe" message. In response, the subscriber receives a WS-Resource-qualified endpoint reference to a "subscription" WS-Resource. The subscription WS-Resource models this relationship between the subscriber and the producer, and it uses WS-ResourceProperties and WS-ResourceLifetime to help manage this relationship.

The second WS-Notification specification, WS-Topics, presents an XML description of topics and associated meta data. Topics are a mechanism for organizing notification messages so that subscribers can conveniently understand what types of notification are available for subscription. Topics can be organized hierarchically; one topic can be further decomposed with child topics. Topics are also scoped by namespace, much as XML types and elements are scoped by XML namespaces.

The third WS-Notification specification, WS-BrokeredNotification, defines the interface to a NotificationBroker that implements an intermediary service to manage subscriptions for other entities in the system that produce notification messages.

9 Conclusions

We have presented the WS-Resource framework, a set of Web service specifications and conventions designed to standardize representation of, and access to, stateful resources in a distributed environment. This framework identifies and standardizes the patterns by which state is represented and manipulated, so that a Web service can describe the stateful resources to which it provides access, and a service requestor can discover the type of this pairing of Web service and stateful resource ("WS-Resource") and use standardized operations to read, update, and query values of its state, and to manage its lifecycle.

The definition of the WS-Resource framework facilitates the construction and use of interoperable services, by making it possible for different service providers and service consumers to describe, access, and manage their stateful resources in standard ways. Equally importantly, the framework introduces support for stateful resources without compromising the ability to implement Web services as stateless message processors. The framework also addresses issues of renewable references, grouping, notification, and fault reporting.

Separate specification documents, summarized in Table 1, provide the normative definition of the concepts presented here.

10 Acknowledgements

This paper has been developed as a result of joint work with many individuals and teams. The authors wish to acknowledge contributions from many people, including:

Nick Butler, Christine Draper, Sonny Fulkerson, Rob High, Jim Knutson, Frank Leymann, Tom Maguire, Susan Malaika, Martin Nally, Jeff Nick, Chris Sharp, Tony Storey, Jay Unger, and Sanjiva Weerawarana. We also acknowledge those with whom we have discussed issues addressed in this paper, including Malcolm Atkinson, Carl Kesselman, and Savas Parastatidis.

11 References

[Design Patterns]

Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns © 1995, Addison Wesley.

[OGSI-Refactor]

Czajkowski, K., Ferguson, D., Foster, I., Frey, J., Graham, S., Snelling, D., Tuecke, S., From Open Grid Services Infrastructure to Web Services Resource Framework: Refactoring and Evolution, 2004. http://www-106.ibm.com/developerworks/webservices/library/ws-resource/gr-ogsitowsrf.html

[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

[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-BaseNotification]

This specification describes the interfaces to basic notification within the WS-Notification family of specifications. It defines the interfaces for the notification producer, notification consumer roles and an interface to manage subscription stateful resources.

ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-BaseN.pdf

[WS-BrokeredNotification]

This specification describes the interface to a notification broker intermediary that rounds out the WS-Notification family of specifications.

ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-BrokeredN.pdf

[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-Resource]

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

[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-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.

[WS-Topics]

This specification describes an XML model for topics and topic spaces within the WS-Notification family of specifications.

ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-Topics.pdf