# WSRF Application Notes

## Committee Draft 02, 25 March 2006

**Document identifier: wsrf-application_notes-1.2-notes-cd-02**

**Location:**

http://docs.oasis-open.org/wsrf/wsrf-application_notes-1.2-cd-02.pdf

**Editors:**

Katy Warr <katy_warr@uk.ibm.com>

Roger Menday <r.menday@fz-juelich.de>

**Abstract:**

This document describes how applications might use the Web Services Resource Framework (WSRF) family of specifications.

**Status:**

This document is published by this TC as a "committee draft".

Committee members should send comments on this specification to the wsrf@lists.oasis-open.org list. Others may submit comments to the TC via the web form found on the TC's web page at http://www.oasis-open.org/committees/wsrf. Click the button for "Send A Comment" at the top of the page. Submitted comments (for this work as well as other works of that TC) are publicly archived and can be viewed at http://lists.oasis-open.org/archives/wsrf-comment/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the WSRF TC web page (http://www.oasis-open.org/committees/wsrf/).

# Table of Contents

# 1 Introduction

This document presents non-normative information that may be of benefit to WS-Resource application developers.

The purpose of this document is to answer common questions that might arise during WSRF application development by means of non-normative scenarios and examples. Additionally, this document may also serve as an aid to clarify potential usage scenarios of the Web Services Resource Framework (WSRF). The intended audience of this document is WS-Resource application designers

This document is divided into sections, each addressing a different aspect of WS-Resource application development. Each section contains information on best practices, along with examples, or references to examples elsewhere.

# 2 Who Should Read This Document?

This Application Notes (AppNotes) document provides a guide for WSRF application developers who already understand the fundamentals of developing a WSRF application. The AppNotes is likely to be used as a reference document in order to answer “frequently asked questions” that might arise during application development. For example: How should multiple WSRF portTypes be combined to produce a single derived portType? Where there is no definitive answer to a question, the AppNotes provides a recommended best practice. This document assumes familiarity with the WSRF specifications and the examples introduced in the [**WSRFPrimer**].

This document does not present a full end-to-end WSRF example and is not meant as an entry point to WSRF. For a WSRF tutorial, readers should refer to the [**WSRFPrimer**].

This document deals first with implementation considerations relating to resource properties [**WS-ResourceProperties**] since the presence of a resource properties document is a defining characteristic of a WS-Resource. Implementation considerations for the remaining WSRF specifications follow. For normative descriptions of WSRF, readers should refer to the WSRF set of specifications: [**WS-Resource**], [**WS-ResourceProperties**], [**WS-ResourceLifetime**], [**WS-BaseFaults**] and [**WS-ServiceGroup**].

# 3 AppNotes Relating to Resource Properties

## 3.1 Defining a Derived WS-Resource in WSDL 1.1

A designer of a WS-Resource application may need to derive a WS-Resource portType by extending or aggregating one or more existing WS-Resource portType(s). The newly created portType will also have an associated resource properties document that may be derived from the existing portType(s). There is a trade-off in portType design between the freedom of the designer to design a portType as he/she wishes and the ability to extend the portType in future (which may place restrictions on the way that a resource properties document is assembled).

### 3.1.1 Best Practice and Examples

#### 3.1.1.1 Resource Properties and Interface Aggregation

Web service interface designers may define a collection of discrete interfaces (portTypes), each of which defines a set of message exchange patterns (operations). A common design scenario is one in which the designer combines these discrete interfaces to form a composed, most-derived interface of a Web service. Examples of independently-specified interfaces designed for purposes of aggregation into a most-derived interface include portTypes defined by WS-ResourceProperties [**WS-ResourceProperties**], WS-BaseNotification [**WS-BaseNotification**], WS-ResourceLifetime [**WS-ResourceLifetime**], and a large number of general-purpose or application-domain-specific management interfaces. Further, there may be various dependencies between these interfaces. For example, the NotificationProducer portType publishes a list of subscribable topics as a resource property, accessed using the operations from the WS-ResourceProperties portType.

Within WSDL 1.1, there is no formally-defined interface extension mechanism[1]. In WSDL 1.1 we expect service designers to copy-and-paste operations from the various constituent interfaces into a single, flat, most-derived service interface. In addition, we expect the service interface designer to compose a resource properties document for the most-derived Web service interface that consists of all resource property element declarations from each of the constituent interfaces used in the composition.

In the following example, a designer wishes to extend the Printer portType described in [**WSRFPrimer**]. The particular portType enables clients to print and create print jobs. An extract from the Printer schema and WSDL follows:

[1] WSDL 2.0 is expected to define a mechanism to formally model interface aggregation /interface/@extends [WSDL 2.0].

```
<xsd:schema
  xmlns:pr="http://docs.oasis-open.org/wsrf/Printer.xsd"
  targetNamespace="http://docs.oasis-open.org/wsrf/Printer.xsd"
  ...>
      ...
      <xsd:element name="PrinterRP">
        <xsd:complexType>
         <xsd:sequence>
           <xsd:element ref="pr:printer_reference" />
           <xsd:element ref="pr:printer_name" />
           <xsd:element ref="pr:printer_state" />
           <xsd:element ref="pr:printer_is_accepting_jobs" />
           <xsd:element ref="pr:queued_job_count" />
           <xsd:element ref="pr:document_format_supported" />
           <xsd:element ref="pr:job_hold_until_default"
                     minOccurs="0"/>
           <xsd:element ref="pr:job_hold_until_supported"
                     minOccurs="0" maxOccurs="unbounded" />
           <xsd:element ref="wsrf-rp:QueryExpressionDialect"
                     minOccurs="0" maxOccurs="unbounded" />
           <xsd:element ref="pr:job_properties"
                     minOccurs="0" maxOccurs="unbounded" />
         </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
</xsd:schema>
```

```
<wsdl:definitions …
  xmlns:prw=http://docs.oasis-open.org/wsrf/Printer.wsdl
  xmlns:pr=http://docs.oasis-open.org/wsrf/Printer.xsd
...>
  <wsdl:types>
    <xsd:schema
 targetNamespace="http://docs.oasis-open.org/wsrf/Printer.xsd" ... >
      <xsd:import
  namespace="http://docs.oasis-open.org/wsrf/Printer.xsd" ... />
    </xsd:schema>
  </wsdl:types>
  ...
  <!-- Association of resource properties document to a portType -->
  <wsdl:portType name="Printer"
             wsrf-rp:ResourceProperties="pr:PrinterRP">

    <!—Operations supported by the Print portType -->
    <wsdl:operation name="Print_Job">  ... </wsdl:operation>
    <wsdl:operation name="Create_Job"> ... </wsdl:operation>

    <!—- WSRF operations supported by this portType -->
    <wsdl:operation
name="GetResourcePropertyDocument">...</wsdl:operation>
    <wsdl:operation name="GetResourceProperty">...</wsdl:operation>
    <wsdl:operation
name="GetMultipleResourceProperties">...</wsdl:operation>
    <wsdl:operation
name="QueryResourceProperties">...</wsdl:operation>
    <wsdl:operation
name="SetResourceProperties">...</wsdl:operation>

  </wsdl:portType>
```

```
</wsdl:definitions>
```

The designer wishes to extend the Printer[2] portType to cater for documents defined by URI. Thus, we require support for the following two new operations:

- Print_URI – Submit a document for printing – document identified by the URI.
- Send_URI – Add to a multi-document job – document identified by the URI.

The new portType will be called URI_Printer. It extends the more generic Printer portType.

The procedure for the derivation of this new portType is an example of manual interface aggregation in WSDL 1.1, using copy-and-paste. A recommendation for the derivation is as follows:

- Define the new portType.

  In this example the new portType is named “URI_Printer”. This portType extends “Printer”.

- Copy all of the operation child elements from the portType being extended, and paste them as child elements of the new portType; the order of the operations should be preserved.

  In this example, the “Print_Job” and “Create_Job” operations are copied from the Printer portType and pasted as child elements of the URI_Printer portType.

- Define new additional operations as child elements of the new portType. When one is extending an already-defined portType, this simply requires a copy-and-paste into the new portType.

  In this example, the “Print_URI” and “Send_URI” operations are new operations defined by the URI_Printer portType.

- Define a new resource properties document, as an XML global element declaration, following the requirements defined in [**WS-ResourceProperties**].

  In this example, the element is named “uri_printer_property” and defined in a new “http://docs.oasis-open.org/wsrf/URIPrinter.xsd” namespace that imports and extends the“http://docs.oasis-open.org/wsrf/Printer.xsd” namespace.

- Copy all of the resource property elements from the resource properties document of the portType being extended, and paste them as child elements of the new resource properties document; the order of the elements should be preserved. This step must be repeated for each portType that is being extended by this new portType. Any duplicate child elements must be removed.

[2] Note that the Printer portType has already been aggregated with operations and resource properties from WS-ResourceProperties.

In this example, the resource property elements such as “pr:printer_name" or "pr:printer_state” are copied from the Printer resource properties document declaration and pasted to the URI_Printer resource properties document declaration.

- Define any additional resource property elements that are specific to the newly-defined resource properties document type.

The resulting URI_Printer schema and WSDL is as follows:

```
<xsd:schema
xmlns:pr="http://docs.oasis-open.org/wsrf/Printer.xsd"
   xmlns:expr="http://docs.oasis-open.org/wsrf/URIPrinter.xsd"
targetNamespace="http://docs.oasis-open.org/wsrf/URIPrinter.xsd"
...>
      <xsd:import
       namespace=http://docs.oasis-open.org/wsrf/Printer.xsd
       ... />
      <xsd:element name="extended_printer_properties">
        <xsd:complexType>
         <xsd:sequence>
           <xsd:element ref="pr:printer_reference" />
           <xsd:element ref="pr:printer_name" />
           <xsd:element ref="pr:printer_state" />
           <xsd:element ref="pr:printer_is_accepting_jobs" />
           <xsd:element ref="pr:queued_job_count" />
           <xsd:element ref="pr:document_format_supported" />
           <xsd:element ref="pr:job_hold_until_default"
                      minOccurs="0"/>
           <xsd:element ref="pr:job_hold_until_supported"
                      minOccurs="0" maxOccurs="unbounded" />
           <xsd:element ref="wsrf-rp:QueryExpressionDialect"
                      minOccurs="0" maxOccurs="unbounded" />
           <xsd:element ref="pr:job_properties"
                      minOccurs="0" maxOccurs="unbounded" />
           <xsd:element ref="expr:uri_printer_property"/>
         </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
</xsd:schema>
```

```
<wsdl:definitions …
  xmlns:pr=”http://docs.oasis-open.org/wsrf/Printer.wsdl”
  xmlns:expr=”http://docs.oasis-open.org/wsrf/URIPrinter.xsd”
  xmlns:exprw="http://docs.oasis-open.org/wsrf/URIPrinter.wsdl"
  ...>
  ...
  <wsdl:types>
    <xsd:schema targetNamespace="http://docs.oasis-
open.org/wsrf/URIPrinter.xsd" ... >
      <xsd:import
  namespace="http://docs.oasis-open.org/wsrf/URIPrinter.xsd" ... />
    </xsd:schema>
  </wsdl:types>
  ...
  <!-- Association of resource properties document to a portType -->
  <wsdl:portType name="URI_Printer"
         wsrf-rp:ResourceProperties=
              "expr:extended_printer_properties">

    <!—Operations supported by the Printer portType -->
    <wsdl:operation name="Print_Job">  ... </wsdl:operation>
```

```
    <wsdl:operation name="Create_Job"> ... </wsdl:operation>

    <!—- WSRF operations supported by this portType -->
    <wsdl:operation
name="GetResourcePropertyDocument">...</wsdl:operation>
    <wsdl:operation name="GetResourceProperty">...</wsdl:operation>
    <wsdl:operation
name="GetMultipleResourceProperties">...</wsdl:operation>
    <wsdl:operation
name="QueryResourceProperties">...</wsdl:operation>
    <wsdl:operation
name="SetResourceProperties">...</wsdl:operation>

    <!— Operations supported by the URI_Printer portType -->
    <wsdl:operation name="Print_URI">  ... </wsdl:operation>
    <wsdl:operation name="Send_URI"> ... </wsdl:operation>

  </wsdl:portType>

</wsdl:definitions>
```

### 3.1.1.2 Creating a New Port Type by Aggregating Existing PortTypes

In the section above we showed how it is possible to create a new portType by adding operations and resource properties into the definition of a most-derived portType. Sometimes a WS-Resource service designer may need to create a portType based on two or more existing portTypes, where one or more of these existing portTypes may have an associated resource properties document.

For example, we may wish to extend the URI_Printer portType with operations and resource properties from a previously defined AdminPortType. We may also wish to preserve the original source of the operations in the composed portType (see following section). Essentially, the process to follow is the same as above: we just use the already-existing WSDL as input to the copy-and-paste process.

### 3.1.1.3 Establishing Aggregated PortType Derivation

There are scenarios where a consumer of a portType that has been created by aggregating a more generic portType needs to discover this derivation in order to find, and act upon, the WS-Resource that is exposed by the portType.

A recommended pre-emptive solution to this problem is to always include sufficient documentation in WS-Resource design so that the origins of each operation or resource property can easily be established.

In addition, we can use a mechanism defined by the WS-Addressing WSDL binding to help identify which portType an operation is derived from. Following the recommendation already described, copy all the operation child elements from the portType being extended, and paste them as child elements of the new portType.

Specify, in the WSDL, a wsa:Action attribute for each wsdl:input and wsdl:output message. Specifically:

- If the wsa:Action attribute does not appear in the child input or output element, then define the wsa:Action for the child input or output element as:
  [target namespace]/[port type name]/[input/output name].
- If the wsa:Action attribute does not appear in the child fault elements then define the wsa:Action for the child fault elements as:
  [target namespace]/[port type name]/[operation name]Fault:[fault name]

where:

[target namespace] is the /definition/@targetnameNamespace of the port type from which the operation was copied

[port type name] is the /definition/porttype/@name of the portType from which the operation was copied

[input/output name] is the name of the element as defined in WSDL 1.1, section 2.4.5.

[fault name] is the /definition/porttype/operation/fault/@name

For example:

wsa:Action="http://docs.oasis-open.org/wsrf/Printer.wsdl/Printer/[input/output name]"

#### 3.1.1.4 Creating a New PortType by Aggregating Existing PortTypes which have a Resource Property with the Same Name but Different Multiplicity

There may be occasions when a WS-Resource service designer creates a portType based on two or more existing portTypes whose resource properties documents contain a property with the same QName.

This scenario presents no problem as long as the semantics (in particularly, the multiplicity) of the shared property are the same in both cases. However, if the semantics of this property differ between the existing portTypes, there may be ambiguity regarding the derived property's behavior. For example, the property is defined as mandatory in one of the derived-from portTypes, and optional in the other.

The best solution to this problem is to revisit the design of the derived-from portTypes and consider why the same property occurs in each with different multiplicity. The portTypes should be restructured to remove the multiplicity conflict.

If this is not possible, (i.e. the derived-from portTypes cannot be changed), a judgment will need to be made. For example, the application designer may choose to take the least restrictive of the conflicting multiplicities.

## 3.2 Defining a Resource Property that Must Always Exist

An attempt to delete a resource property whose minOccurs>0, using the DeleteResourceProperties message exchange pattern (MEP) (or the DeleteResourceProperties component of a SetResourceProperties MEP), will result in the InvalidModificationFault because deletion of this property would render the resource properties document invalid.

## 3.3 Invalid or Non-existent Resource Properties

This section describes how to distinguish between properties that are valid, invalid, or unavailable, and those that are available but have no current value assigned.

### 3.3.1 Best Practice and Examples

#### 3.3.1.1 Establishing a List of Valid Resource Properties

A client wishing to establish whether a resource property is present in a resource property document should use the GetResourceProperty request on a resource, passing the property name in question. Since resource properties may be dynamically inserted into or deleted from a resource properties document, the returned list of valid resource properties may vary for a particular document over time. Additionally, if the resource properties schema allows open content via an xsd:any element, then even the list of valid resource property QNames may vary in a particular document over time.

A client wishing to establish all the resource properties that are present for a resource should issue the GetResourcePropertiesDocument method on the resource.

An application developer may consider using an additional resource property to publish a list of QNames of resource properties that are valid and/or present, for a resource that supports open content in its resource properties document. In many cases this may prove more economical than retrieving the entire resource properties document.

#### 3.3.1.2 Invalid Properties

When a GetResourceProperty operation returns an InvalidResourcePropertyQNameFault, the property requested does not exist in the resource properties document schema for the WS-Resource.

Similarly, a client receives an InvalidResourcePropertyQNameFault in response to a GetMultipleResourceProperties request when one or more of the resource property QNames in the request message do not correspond to resource properties in the resource property document schema.

#### 3.3.1.3 Properties That Are Valid but Have No Value

If a client issues GetResourceProperty on a property described in the schema with minOccurs='0', the GetResourceProperty operation returns an empty GetResourcePropertyResponse message if there is currently no resource property of this type in the resource properties document.

Similarly, the GetMultipleResourceProperties operation returns a collection of the properties corresponding to the QNames in the request message. In the case where a resource properties document does not contain a value for one of the requested properties, no element is added to the collection for that property's QName.

#### 3.3.1.4 Properties That Have the Value 'nil'

If a property is declared nillable and it has the value nil, the GetResourceProperty operation will return the resource property element decorated with an xsi:nil="true" attribute.

Similarly, the GetMultipleResourceProperties operation returns a collection of the properties corresponding to the QNames in the request message. Any property whose value is nil will be decorated with the xsi:nil="true" attribute.

The TerminationTime resource property defined in [**WS-ResourceLifetime**] is an example of a nillable Resource Property. A service requestor can specify that there is no scheduled termination time for the WS-Resource by using the SetResourceProperty operation to set the value of this Resource Property to nil:

```
<wsrl:TerminationTime xsi:nil="true"/>.
```

## 3.4 Resource Property Attributes

WS-Resources may associate attributes with individual resource property definitions in order to indicate that instances of that resource property will exhibit a particular behavioral trait. This section explains how this is done and gives some examples illustrating why it might be useful.

### 3.4.1 Best Practice and Examples

#### 3.4.1.1 Refining a Resource Property Definition with Lifetime Attributes

Consider a WS-Resource that represents an item in a warehouse. The item may have a price associated (by means of a resource property) and, on occasion, a sale price. In this example,

the resource property defining the sale price is only valid between specific dates, so it is necessary to indicate to the consumer some lifetime aspects of this property. In this case GoodFrom and GoodUntil are attributes added to the sale price resource property to indicate the date that the sale price came into effect and the date until when it is valid. Attributes such as GoodFrom and GoodUntil can appear on a resource property if:

- the resource property definition explicitly includes them as attributes

or if:

- the resource property definition allows attribute extensibility by associating anyAttribute with the property definition (as described in the example in section 3.4.1.2, Providing Attribute Extensibility to a Resource Property).

The definition of the sale price resource property with GoodFrom and GoodUntil explicitly defined as attributes is illustrated by the following XML schema:

```
<xsd:element name="SalePrice">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="GoodFrom" type="xsd:dateTime"/>
        <xsd:attribute name="GoodUntil" type="xsd:dateTime"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

The following is an example SalePrice resource property:

```
<SalePrice
  GoodFrom="2006-01-23T09:59:55.025+01:00"
  GoodUntil="2006-01-23T12:59:55.093+01:00" …>
    546.33
</SalePrice>
```

### 3.4.1.2 Providing Attribute Extensibility to a Resource Property

The CurrentTime resource property defined in [**WS-ResourceLifetime**] is an example of a Resource Property definition that allows attribute extensibility by associating anyAttribute with the property definition.

The XML schema definition for the CurrentTime resource property element type is as follows:

```
<xsd:element name="CurrentTime">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:dateTime">
        <xsd:anyAttribute namespace="##other"
                          processContents="lax"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

The xsd:anyAttribute in this resource property enables application designers to associate an attribute that has not been pre-specified with the CurrentTime. An example might be an attribute indicating the accuracy of the CurrentTime property.

## 3.5 Adding Attributes to the Root of the Resource Properties Document

No restrictions are placed on the adding of attributes to the resource properties document root. Consumers of the WS-Resource can retrieve attributes on the resource properties document root by exploiting the GetResourcePropertyDocument MEP.

For example, the PrinterRP resource properties document declaration introduced earlier might be extended as follows to introduce a mandatory attribute to record the creation date of the printer resource:

```
<xsd:element name="PrinterRP">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="pr:printer_reference" />
      <xsd:element ref="pr:printer_name" />
      …
    </xsd:sequence>
    <xsd:attribute name="CreationDate" type="xsd:dateTime"
                                            use="required"/>
  </xsd:complexType>
</xsd:element>
```

An extract of the resource property document instance is:

```
<PrinterRP CreationDate="2006-01-23T12:25:54.507+01:00" ...>
  …
</PrinterRP>
```

In order to query attributes on the root of the resource properties document, a WS-Resource client may extract the complete resource properties document using the GetResourcePropertyDocument MEP and query the specific attribute(s). It is also possible to use the QueryResourceProperties operation (when available) forming a query which reads the root attributes[3].

Similarly, to perform a resource properties document root attribute update, the client application must update the attribute in a local copy of the resource properties document and then update the resource properties document associated with the resource, using the PutResourcePropertyDocument MEP.

## 3.6 Notifying Clients of Resource Property Changes

A common pattern for WS-Resource applications is for the WS-Resource to send automatic notifications of changes to resource property elements of its resources to interested partners.

[3] We refer the reader to section 8.2 for interoperability recommendations regarding namespace prefixes and XPath queries.

Resource properties may be changed by external events (for example, by use of SetResourceProperties) or by events internal to the service and not directly visible to the client (for example, a printer is put out of service).

## 3.6.1 Best Practice and Examples

### 3.6.1.1 Resource Property Value-Change Notification Pattern

In order to provide automatic notification of resource property changes, application designers should compose WSRF with [**WS-BaseNotification**].

A normative description of how this should be done is contained in [**WS-ResourceProperties**], and an example is contained in the [**WSRFPrimer**].

### 3.6.1.2 Format of the Resource Property Value-Change Notification

The notification messages can take many forms. The application developer should be aware that the ResourcePropertyValueChangeNotification element may occur anywhere within the notification message. Likewise, a notification subscriber should ensure that a notification consumer is able to process the particular form of the notification message sent by the producer.

The element may appear as the root element of the notification message, or as a descendent of the root element, accommodating patterns where the notification message is contained in an enveloping mechanism. For example, Web Services Distributed Management [**WSDM**] defines a format for management event messages, which may then embed a ResourcePropertyValueChangeNotification element.

# 4 AppNotes Relating to Lifecycle

## 4.1 Resource Lifecycle Management

This section describes how individual resources that are fronted by a WS-Resource Web service are created and destroyed.

### 4.1.1 Best Practice and Examples

#### 4.1.1.1 Resource Creation

The specifications in the WSRF family do not specify how a WS-Resource instance should be created. A commonly used pattern is the “factory pattern”, whereby a separate Web service (factory) exposes an operation to clients for creating a new resource instance and returning a reference to that instance.

The explicit factory pattern is by no means the only mechanism by which resource instances might be created. A resource might be created as part of an operation performing a wider function. For example, in WS-Notification, a subscription request creates an endpoint to represent the subscription and returns the relevant endpoint reference (EPR) as part of its behavior.

Refer to the [**WSRFPrimer**] for examples of resource creation, such as the creation of a print job when a document is printed.

#### 4.1.1.2 Resource Destruction

The WS-Resource Lifetime specification defines the mechanisms by which the lifecycles of resources should be managed. Resources can be explicitly destroyed or scheduled for destruction. Refer to [**WS-ResourceLifetime**] for details.

A common practice is to specify the resource's initial termination time as part of its creation – thus saving an operation.

# 5 AppNotes Relating to Base Faults

## 5.1 Fault Handling

Problem determination is an important aspect of Web service application development. This section describes how best to generate faults in the event of an error, and how a client of a Web service might process the faults in order determine the underlying problem.

This section describes best practices for fault generation and consumption in a WS-Resource application environment.

### 5.1.1 Best Practice and Examples

A WS-Resource application should adhere to the [**WS-BaseFaults**] specification for all its fault processing. Adhering to this standard has a number of benefits:

- The [**WS-BaseFaults**] model removes the need for proprietary or application-specific fault handling. Fault recipients may therefore be developed in isolation from the service generating the fault.
- [**WS-BaseFaults**] provides a simple and powerful pattern for fault processing, enabling the application designer to focus on the design of the application rather than the underlying fault processing model.
- A standard model for fault processing enables re-use of code both in the recipient of the fault and in the fault generator. This also eases the development of tools.

Faults are generated either in response to errors in the Web service application (service faults) or as a result of some kind of system processing error (system faults). For example, a system fault may be generated as part of transport processing. This section deals with the handling of service faults, but it is worth noting that a service might raise a service fault as a result of an underlying system fault cause.

#### 5.1.1.1 Defining and Generating Base Faults

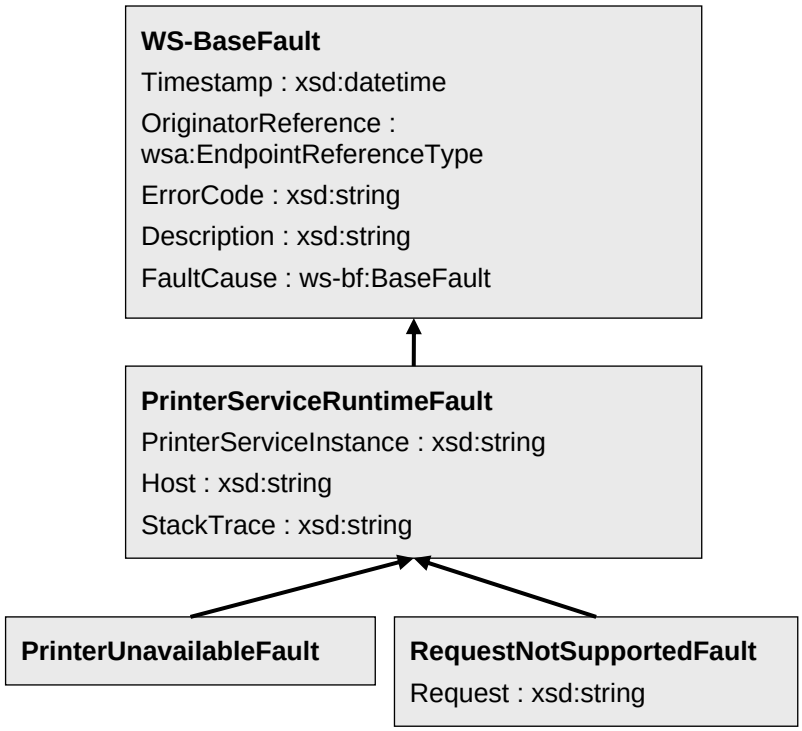
Each base fault that might be generated by a service requires its own distinct XML schema type that extends wsrf-bf:BaseFault. This extended fault complexType may contain additional attributes and/or elements.

Clearly, the type of information that should be defined in an extended fault type depends on the application and its deployment. Ensuring that the correct information is available in the fault is critical to effective problem determination. Here are some recommended additional elements that might be contained in the extended fault:

- Host: The host on which the fault was generated. If the service could be run on more than one host (for example, in the case of load balancing), it is important to include the host name as an element in the extended fault type.
- Process: The process in which the service was running when the fault occurred.
- SOAPFault information: If the WS-BaseFault is to wrap a SOAP fault then the SOAP fault code and role should be contained in the fault.

Consider the Printer example used previously. We require extended fault types for use by our printer portType. To begin, it would be prudent to define a generic extended fault for the printer service from which all other printer service faults could be derived. This fault type will define the basic information that we require in every fault generated by the printer service. For example, every fault generated should contain an identifier specifying the printer to which the fault was directed, the host, and a stack trace detailing the fault cause. Further extended

584 faults may be generated from this underlying generic type (for example:
585 PrinterUnavailableFault and RequestNotSupportedFault).
586




587
588
589 An example of a PrinterServiceRuntimeFault SOAP 1.1 message follows:

```
590 <s11:Envelope
591     xmlns="http://schemas.xmlsoap.org/soap/envelope/"
592     xmlns:s11="http://schemas.xmlsoap.org/soap/envelope/"
593     xmlns:wsa=" http://www.w3.org/2005/08/addressing"
594     xmlns:wsrf-bf="http://docs.oasis-open.org/wsrf/bf-2"
595     xmlns:wsrf-printer="http://docs.oasis-open.org/wsrf/printer">
596   <s11:Header>
597     <wsa:Action>
598       http://docs.oasis-open.org/wsrf/fault
599     </wsa:Action>
600     …
601   </s11:Header>
602   <s11:Body>
603     <s11:Fault>
604       <faultcode>s11:Client</faultcode>
605       <faultstring>No such resource exists</faultstring>
606       <faultactor>http://example.org/someactor</faultactor>
607       <detail>
608         <wsrf-printer:PrinterServiceRuntimeFault>
609           <wsrf-bf:Timestamp>
610             2005-05-04T20:18:44.970Z
611           </wsrf-bf:Timestamp>
612           <wsrf-printer:Host>
613               123.434.434.33
614           </wsrf-printer:Host>
615         </wsrf-printer:PrinterServiceRuntumeFault>
616       </detail>
617     </s11:Fault>
618   </s11:Body>
```

```
</s11:Envelope>
```

In the example above, the PrinterServiceInstance and Host elements are not present within the PrinterServiceRuntimeFault as they are optional.

#### 5.1.1.2 Relaying Base Faults

Faults received by an intermediary should be wrapped by a fault containing the intermediary information and then relayed. The wrapped fault is contained in the FaultCause element of the new enclosing fault.

# 6 AppNotes Relating to ServiceGroup

## 6.1 ServiceGroup Members with Derived PortTypes

Often a ServiceGroup [**WS-ServiceGroup**] will be constructed which contains member services with derived portTypes (see section 3.1 for derivation procedure guidelines). This section clarifies the application of the ServiceGroups specification for services with derived portTypes.

### 6.1.1 Best Practices and Examples

The MembershipContentRule resource property element of a ServiceGroup defines the conditions that must be met by a ServiceGroupEntry of a ServiceGroup. A ServiceGroupEntry may be restricted by the MemberInterfaces and ContentElements attributes of the MembershipContentRule.

#### 6.1.1.1 MemberInterfaces Attribute for Derived PortTypes

The ServiceGroup specification refers only to normatively-defined information in other specifications. Therefore, the most derived WSDL 1.1 portType is the name to be used in the MemberInterfaces attribute of the MembershipContentRule. For example, from the derived Printer portType of section 3.1.1.1, we could specify the following MembershipContentRule:

```
<wsrf-sg:MembershipContentRule MemberInterfaces="prw:URI_Printer" />
```

#### 6.1.1.2 ContentElements Attribute for Derived PortTypes

It may be convenient to build a ServiceGroup whose member services have some aspect of common behavior which is defined by the inclusion of operations and properties from some common component portType.

Using the behaviors defined by the ScheduledResourceTermination portType as an example, the ServiceGroup may include the following MembershipContentRule:

```
<wsrf-sg:MembershipContentRule
            ContentElements="wsrf-rl:TerminationTime" />
```

This constrains the content of every ServiceGroupEntry to require a TerminationTime element. The service built on ServiceGroup should then require the following (which are not normative requirements of the ServiceGroup specification):

1. The corresponding member service also has the TerminationTime property in its resource properties document schema
2. The SetTerminationTime operation exists in the portType of the member service.
3. The WSDL of the member service includes the wsa:Action attribute in the input element of this operation. The attribute value is http://docs.oasis-open.org/wsrf/rlw-2/ScheduledResourceTermination/SetTerminationTimeRequest.

These three requirements relate the content of the ServiceGroupEntry to behavior defined by the ScheduledResourceTermination portType.

# 7 AppNotes Relating to Resource Metadata

## 7.1 Derived Resource Metadata

Resource Metadata [**WS-ResourceMetadataDescriptor**] specifies a format for additional metadata which describes the resource properties of a WS-Resource. As with derived portTypes using “cut-and-paste” a similar process should be followed for the derived MetadataDescriptor document.

### 7.1.1 Best Practices and Examples

A MetadataDescriptor component can optionally specialise one or more other MetadataDescriptor components. However, as with WSDL 1.1, there is no formal way to extend an existing MetadataDescriptor without copying its contents into the new MetadataDescriptor. In such cases the MetadataDescriptor component should contain the properties of the MetadataDescriptor component(s) it specializes in addition to the properties it already defines.

When performing this aggregation, it is important to remember that no two Property elements can share a {path} value. If a new MetadataDescriptor extends two other MetadataDescriptors that happen to share a Property {path}, the new descriptor must consolidate these duplicate Property definitions into one. The author of the new descriptor is free to decide which of the Property attributes and child elements will be included in the new definition, to prevent the scenario described above (because we are using cut-and-paste rather than formal inheritance, the author is not obliged to honour the restrictions of previous definitions).

# 8 Interoperability Considerations

## 8.1 Interoperability

In order to ensure interoperability of WSRF implementations, WSRF applications should use a document-literal binding to serialize the messages defined by the WSRF specifications.

## 8.2 Intermediary Changes of Namespace Prefixes

SOAP allows intermediaries to modify the XML namespace prefixes of messages passing through them. This may result in interoperability problems if attributes or text nodes in the message contain `QNames`, as the `QName` prefixes in attributes and text nodes will not be recognized as such by the intermediary. For example, the following query uses the prefix 'pr' to refer to a component of the Printer properties in order to discover whether the printer supports the 'text/plain' MIME type:

```
<s11:Body>
   <wsrf-rp:QueryResourceProperties>
    <wsrf-rp:QueryExpression
        xmlns:pr=http://docs.oasis-open.org/wsrf/Printer.xsd
        Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
                contains(/*/*/pr:mimeMediaType,"text/plain")
     </wsrf-rp:QueryExpression>
    </wsrf-rp:QueryResourceProperties>
</s11:Body>
```

### 8.2.1 Best Practices and Examples

For cases where the prefix is in an XPath, the XPath expressions can be rewritten in such a way that they are protected against such intermediary behavior. For example, the following query uses the full namespace of the Printer property:

```
<s11:Body>
   <wsrf-rp:QueryResourceProperties>
    <wsrf-rp:QueryExpression
        Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
                contains(/*/*/*[namespace-uri()=
      'http://docs.oasis-open.org/wsrf/Printer.xsd'
      and local-name()='mimeMediaType'],"text/plain")
     </wsrf-rp:QueryExpression>
    </wsrf-rp:QueryResourceProperties>
</s11:Body>
```

Alternatively, an application deployer might choose to use a different vendor's intermediary that does not alter the message prefixes.

# References

## 8.3 Normative


**[WS-Addressing]**
http://www.w3.org/TR/ws-addr-core

**[WS-BaseFaults]**
http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf

**[WS-BaseNotification]**
http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-pr-02.pdf

**[WS-Resource]**
http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf

**[WS-ResourceLifetime]**
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-os.pdf

**[WS-ResourceMetadataDescriptor]**
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_metadata_descriptor-1.0-spec-cd-01.pdf

**[WS-ResourceProperties]**
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf

**[WS-ServiceGroup]**
http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-spec-os.pdf

**[WS-Topics]**
http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-pr-01.pdf


## 8.4 Non-Normative


**[WSRFPrimer]**
http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf

**[WSDM]** Web Services Distributed Management
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm

# Appendix A. Acknowledgments

The following individuals were members of the committee during the development of this document:

Mario Antonioletti (EPCC, The University of Edinburgh), Akhil Arora (Sun Microsystems), Tim Banks (IBM), Jeff Bohren (OpenNetwork), Fred Carter (AmberPoint), Martin Chapman (Oracle), Glen Daniels (Sonic Software), David De Roure (University of Southampton), Thomas Freund (IBM), John Fuller (Individual), Stephen Graham (IBM), Anish Karmarkar (Oracle), Hideharu Kato (Hitachi), David Levine (IBM), Paul Lipton (Computer Associates), Mark Little (Arjuna Technologies Limited), Lily Liu (WebMethods, Inc.), Tom Maguire (IBM), Susan Malaika (IBM), Mark Mc Keown (University of Manchester), David Martin (IBM), Samuel Meder (Argonne National Laboratory), Jeff Mischkinsky (Oracle), Roger Menday (Forschungszentrum Juelich GmbH), Bryan Murray (Hewlett-Packard), Mark Peel (Novell), Alain Regnier (Ricoh Company, Ltd.), Ian Robinson (IBM), Tom Rutt (Fujitsu), Mitsunori Satomi (Hitachi), Igor Sedukhin (Computer Associates), Hitoshi Sekine (Ricoh Company, Ltd.), Frank Siebenlist (Argonne National Laboratory), Alex Sim (Lawrence Berkeley National Laboratory), David Snelling (Fujitsu), Latha Srinivasan (Hewlett-Packard), Rich Thompson (IBM), Jem Treadwell (Hewlett-Packard), Steve Tuecke (Argonne National Laboratory), William Vambenepe (Hewlett-Packard), Katy Warr (IBM), Alan Weissberger (NEC Corporation), Pete Wenzel (SeeBeyond Technology Corporation), Kirk Wilson (Computer Associates) and Umit Yalcinalp (SAP).

# Appendix B. Revision History

| Rev | Date (MM/DD/YYYY) | By Whom | What |
|---|---|---|---|
| 0.1 | 11/07/2005 | Katy Warr | Initial Creation based on AppNotes outline from Alan |
| 0.2 | 16/01/2005 | Katy Warr | Begin to bring in line with Primer and other docs for consistency and to remove duplication. |
| 0.3 | 02/01/2005 | Katy Warr | Updated base fault section |
| 0.4 | 21/02/2005 | Katy Warr | Minor changes for f2f |
| 0.5 | 03/09/2005 | Katy Warr | Some of the issues raised at the feb f2f:<br>- Add updates to do section so discussions aren't forgotten.<br>- moved non-normative txt from WSRP section 4 to appnotes<br>- corrections and minor amendments<br>- improve RAP embodiments section<br>- Remove QOS section |
| 0.6 | 05/16/2005 | Katy Warr | Draft resolutions to the following issues:<br>22, 63, 52, 89, 95 |
| wd-03 | 05/20/2005 | Ian Robinson | Editorial consistency |
| pr-01 | 06/13/2005 | Ian Robinson | Change status to PR draft |
| wd-04 | 10/18/2005 | Roger Menday | Resolution to 103. Misc. edits. Replaced DiskDrive RP example with Printer. Integrated feedback. |
| wd-05 | 11/28/2005 | Roger Menday | Integrated comments from Ian Banks, Jem Treadwell and Daniel Jemiolo. |
| wd-06 | 1/18/2006 | Roger Menday | New sections on ServiceGroup and Resource Metadata. |
| wd-07 | 2/18/2006 | Roger Menday | Integrated comments from Ian Robinson and Jem Treadwell. |
| cd-02 | 3/25/2005 | Roger Menday | Inserted some post-vote edits from Kirk Wilson. |

# Appendix C. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.



This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.