WSRF Application Notes

Working Draft 03, 20 May 2005

Document identifier: wsrf-application_notes-1.2-notes-cd-01.doc

Location:
http://docs.oasis-open.org/wsrf/wsrf-application_notes-1.2-notes-cd-01.pdf

Editors:
Katy Warr <katy_warr@uk.ibm.com>

Abstract:
This document describes how applications might use the WS-RF family of specifications.

Status:
This document is published by this TC as a "comittee draft". It is possible that it may change during this process, but should nonetheless provide a stable reference for discussion and early adopters' implementations.

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 WS-RF 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 WS-RF. The intended audience of this document are 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 reference to examples elsewhere.

2 Who Should Read This Document?

This document provides a guide for WS-RF application developers who already understand the fundamentals of developing a WS-RF application. The AppNotes is likely to be used as a reference document in order to answer common questions that might arise during application development. For example: *How should multiple WS-RF 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 WS-RF specifications and the examples introduced in the [WSRFPrimer].

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

For normative descriptions of WS-RF, readers should refer to the WS-RF set of specifications: [WS-Resource], [WS-ResourceProperties], [WS-ResourceLifetime], [WS-BaseFaults] and [WS-ServiceGroup].

3 AppNotes relating to Resource Properties

3.1 Defining a 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 associated resource property 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 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 WS-Notification [WS-Notification], 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. That is, the messages defined by interface A may only be useful in a service implementation when combined with those of interface B.

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 property document for the most-derived Web service interface that consists of all of the resource property element declarations from each of the constituent interfaces used in the composition.

Consider the following example, wherein a designer extends the “GenericDiskDrive” WS-Resource interface in a vendor-specific fashion.

```
<wsdl:definitions …
    xmlns:gen="http://example.com/diskDrive"
    xmlns:ven="http://vendor.com/diskDrive"
   …>
…
  <wsdl:types>
    <xsd:schema targetNamespace="http://vendor.com/diskDrive" ... >

      <!-- Resource property element declarations -->
      <xsd:element name="VendorExtension" type="xsd:string" />
```

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

```
      <!-- Resource properties document declaration -->
      <xsd:element name="VendorDiskDriveProperties">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="gen:NumberOfBlocks"/>
            <xsd:element ref="gen:BlockSize" />
            <xsd:element ref="gen:Manufacturer" />
            <xsd:element ref="gen:StorageCapability"
                         minOccurs="0" maxOccurs="unbounded" />
            <xsd:element ref="ven:VendorExtension" />
            <xsd:any minOccurs="0" maxOccurs="unbounded" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
…
    </xsd:schema>
  </wsdl:types>
…
  <!-- Association of resource properties document to a portType -->
  <wsdl:portType name="VendorDiskDrive"
    wsrf-rp:ResourceProperties="ven:VendorDiskDriveProperties" >
    <operation name="…
…
    <!-- copy/paste operations from genericDiskDrive -->
    <operation name="start" …/>
    <operation name="stop" …/>
…
    <!-- define Vendor-specific operations -->
    <operation name="reset" …/>
…
  </wsdl:portType>
…
</wsdl:definitions>
```

The VendorDiskDrive portType is an example of *manual* interface aggregation in WSDL 1.1 using copy-and-paste. In this example, the designer of the VendorDiskDrive portType wishes to *extend* the GenericDiskDrive portType.

WS-ResourceProperties specifies that this style of extension MUST be carried out in the following fashion:

1. Define the new portType.

   In this example the new portType is named “VendorDiskDrive”. This portType extends “GenericDiskDrive”.

2. 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 “start” and “stop” operations are copied from the GenericDiskDrive portType and pasted as child elements of the VendorDiskDrive portType.

3. Define additional, vendor-specific operations as child elements of the new portType.

   In this example, the “reset” operation is a new operation defined by the VendorDiskDrive portType.

4. 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 “VendorDiskDriveProperties” and defined in the “http://vendor.com/diskDrive” namespace.

5. Copy all of the child elements (@ref and xsd:any) 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.

   In this example, the elements that reference (@ref) “gen:NumberOfBlocks”, “gen:Blocksize”, and “gen:Manufacturer”, “gen:StorageCapability” and the “xsd:any” are copied from the GenericDiskDriveProperties declaration and pasted to the VendorDiskDriveProperties declaration.

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

   In this example, VendorDiskDriveProperties resource document defines an additional resource property named VendorExtension.

   TO DO: Change the above example to relate to the Primer's printer type example, rather than the vendor disk-drive. Something like:

Consider extending the *Printer portType* described in [WSRFPrimer] to cater for printer documents defined by a URI.

The complete set of *printer-properties* is defined in a properties document at http://???/PrinterResourceProperties.xsd:

```
  <xsd:element name="printer-properties">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="printer-reference"/>
        <xsd:element ref="printer-name"/>
        <xsd:element ref="printer-state"/>
        <xsd:element ref="printer-is-accepting-jobs"/>
        <xsd:element ref="queued-job-count"/>
        <xsd:element ref="operations-supported"/>
        <xsd:element ref="document-format-supported"/>
        <xsd:element ref="job-properties" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

This service exposes a *Printer* portType to enable clients to (for example) print and create print jobs. This PortType is associated with a Resource Properties document in WSDL 1.1 as follows:

```
xmlns:wsrf-pr=
```

```
"http://www.WSRF-
Examples.org/IPPprinters/PrinterResourceProperties.xsd"

  :

  <wsdl:portType name="Printer" wsrf-rp:ResourceProperties="wsrf-
pr:printer-properties">

    <!—Operations supported by the Print PortType -->
    <wsdl:operation name="Print-Job">  ... </wsdl:operation>
    <wsdl:operation name="Create-Job"> ... </wsdl:operation>
    <wsdl:operation name="Send-URI">  ... </wsdl:operation>

    <!-- WS-RF operations supported by this portType -->
    <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>
```

In particular, we require support for the following two 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* and it extends the more generic *Printer* portType.

The first step is to copy all the operation child elements from the Printer portType into the newly defined URI-Printer portType containing the addition operations. This will result in the following WSDL:

```
  <wsdl:portType name="URI-Printer"

    <!—Operations supported by the URI-Printer PortType -->
    <wsdl:operation name="Print-URI"> ... </wsdl:operation>
    <wsdl:operation name="Send-URI"> ... </wsdl:operation>


    <!—Operations from by the Printer PortType -->
    <wsdl:operation name="Print-Job"> ... </wsdl:operation>
    <wsdl:operation name="Create-Job"> ... </wsdl:operation>


    <!-- WS-RF operations supported by this portType -->
    <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>
```

The next step is to copy all of the child elements from the *printer-properties* resource properties document and paste them into a new resource properties document located at http://???/URIPrinterResourceProperties.xsd.

```
  <xsd:element name="uri-printer-properties">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="printer-reference"/>
        <xsd:element ref="printer-name"/>
        <xsd:element ref="printer-state"/>
        <xsd:element ref="printer-is-accepting-jobs"/>
        <xsd:element ref="queued-job-count"/>
        <xsd:element ref="operations-supported"/>
        <xsd:element ref="document-format-supported"/>
        <xsd:element ref="job-properties" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

Finally, the *uri-printer-properties* resource properties document is associated with the *uri-printer* portType and the relevant XML global element declaration is added.

```
xmlns:wsrf-upr="http://???/URIPrinterResourceProperties.xsd"


  <wsdl:portType name="URI-Printer" wsrf-rp:ResourceProperties="wsrf-
upr:uri-printer-properties">
        :
  </wsdl:portType>
```

3.1.1.2 Creating a new Port Type by Aggregating Existing PortTypes

This example is included here to serve as an introduction to the following examples.

A WS-Resource service designer may need to create a PortType based on two or more existing PortTypes. One or more of these existing PortTypes may have an associated resource property document.

Once again, there is not formally-defined interface extension mechanism in WSDL 1.1and the service designer must *copy and paste* the various constituent PortTypes and associated resource properties into a single, flat, most-derived service interface.

The Printer portType was specialized to create a URI-Printer PortType above. Let's assume a second specialization of the Printer PortType to create an AdminPortType. The AdminPortType will expose operations only available to an administrator.

> To do: add the example where AdminPortType is an aggregation of Printer and Admin (which introduces additional operations). This example will be presented in a similar way to the one above.

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.

There is no standard mechanism for establishing a PortType operation's origin. However, this can usually be implied by the operation's name and its semantics. A similar approach can be used in establishing the origin of Resource properties.

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

It should never be possible to derive multiple operations or Resource Properties of the same name from varying sources. Good Web service design practice should ensure that the operation QNames are always unique.

3.1.1.4 Creating a new PortType by Aggregating Existing PortTypes which have a Resource Property with the same Name but different Cardinality

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

This scenario presents no problem as long as the semantics (in particularly, the cardinality) 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 behaviour. An example 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 re-visit the design of the derived-from PortTypes and consider why it is that the same property occurs in each with different cardinality. The PortTypes should be restructured to remove the cardinality conflict.

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

This is a specific example of a more generic problem - that of combining operations of the same name but different semantics from aggregated port types. In order to avoid this problem altogether, the application designer should ensure that operations or properties that are re-used always retain the same semantics.

3.2 Creating a Resource Property document by extending an existing schema

Comment: (ir) Was there a specific issue that motivates this? Is there any reason to suppose a resource property document could *not* be extended in the normal fashion described by XML schema (as illustrated here)?

A Resource Property Document may be created by extending an existing schema by use of xsd:extension. Consider the basic GenericDiskDriveProperties Resource Properties schema:

```
<xsd:element name="GenericDiskDriveProperties">
    <xsd:complexType>
```

```
        <xsd:sequence>
              <xsd:element ref="tns:NumberOfBlocks"/>
              <xsd:element ref="tns:BlockSize"/>
              <xsd:element ref="tns:Manufacturer"/>
              <xsd:any minOccurs="0" maxOccurs="unbounded"/>
              <xsd:element ref="tns:StorageCapability"
                        minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsdcomplexType>
</xsd:element>
```

The above might be extended to include an attribute indicating the creation date as follows:

```
<xsd:element name="GenericDiskDrivePropertiesWithCreationDate">
    <xsd:complexType>
         <xsd:complexContent>
              <xsd:extension base="tns:GenericDiskdriveProperties">
                   <xsd:attribute name="CreationDate"
type="xsd:DateTime" use="required"/>
              </xsd:extension>
          </xsd:complexContent>
    </xsd:complexType>
</xsd:element>
```

Comment: (sgg) Again, an example here would be good.

3.3 Why Resource Property Documents and Resource Properties must be GEDs?

The resource properties document itself must be a GED in some XML namespace. This GED defines the type of the root element of a resource properties document and hence the type of the resource properties document itself. This requirement ensures the resource property document's uniqueness in a particular namespace and provides a mechanism to specify the document (and hence resource properties) associated with a given portType.

Each resource property element must itself be a GED in the resource properties document. Once again, the same principle is behind this restriction: resource property QName uniqueness within the document could not be ensured if the resource properties were not restricted to GEDs.

This restriction ensures that resource properties are referenced in an unambiguous way in Message Exchange Patterns such as `getResourceProperty(QName)`. In addition, the following (invalid) resource property document illustrates how two QNames could clash if the GED restriction was not in place:

Comment: (ir) I'm not sure the example properly illustrates why we have the restriction on GEDs - there is no actual problem with the use of tns:Size as described - the <Size> child of <Blocks> is a part of the <Blocks> RP - it is not an RP in itself. And both <tns:Szie> and <tns:Blocks> are uniquely defined by qnames in this example. The problem would come if a new resource properties document needed to be specified that composed GenericDiskDriveProperties with another resource properties document in a new namespace. A GED allows an element to be easily mixed in without having to reproduce the type definition in the new namespace.

```
<xsd:element name="InvalidGenericDiskDriveProperties">
    <xsd:complexType>
         <xsd:sequence>
              <xsd:element name="tns:Blocks">
                   <xsd:complexType>
                        <xsd:sequence>
                             <xsd:element ref="tns:Number"/>
                             <xsd:element ref="tns:Size"/>
```

```
                            </xsd:sequence>
                        </xsdcomplexType>
                </xsd:element>
                <xsd:element ref="tns:NumberOfBlocks"/>
                <xsd:element ref="tns:Size"/>
                <xsd:element ref="tns:Manufacturer"/>
                <xsd:any minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="tns:StorageCapability"
                             minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsdcomplexType>
</xsd:element>
```

The child element with the reference "`tns:size`" refers to the individual block size. The GED definition also referenced as "`tns:size`" refers to the size of the disk drive. Both these reference properties share the same `QName` and therefore uniqueness has not been enforced.

3.4 Defining a Resource Property that must always exist

A resource property that must always exist must be defined in the XML schema definition for the resource properties document as always having at least one occurrence. This can be achieved by specifying (for example) `minOccurs > 0` or `minOccurs = 1` for the property in question.

There are many examples of resource properties that must always have at least one occurrence. The GenericDiskDriveProperties document includes a property defining the manufacturer. The following example illustrates how the GenericDiskDriveProperties document might be updated to ensure that the manufacturer is a mandatory property and occurs exactly once:

```
<xsd:element name="GenericDiskDriveProperties">
    <xsd:complexType>
        <xsd:sequence>
                <xsd:element ref="tns:NumberOfBlocks"/>
                <xsd:element ref="tns:BlockSize"/>
                <xsd:element ref="tns:Manufacturer"
                         minOccurs="1" maxOccurs="1"/>
                <xsd:any minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="tns:StorageCapability"
                             minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsdcomplexType>
</xsd:element>
```

Note that attempts to delete a resource property whose `minOccurs > 0` using the `DeleteResourceProperties` MEP (or the DeleteResourceProperties component of a SetResourceProperties MEP) will result in the Invalid`Modification` fault because deletion of this property would render the resource property document invalid.

3.5 Invalid or Non-existent Resource Properties

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

3.5.1 Best Practice and Examples

3.5.1.1 Establishing a list of Valid Resource Properties

A client wishing to establish whether a resource property is defined at this time should use `GetResourceProperty` request on a resource, passing the property in question. As resource properties may be dynamically inserted/deleted to a Resource Property document containing `xsd:any`, the returned list of valid resource properties may vary for a particular document over time.

Similarly, a client wishing to establish all the Resource Properties that are available for a resource should issue the `getResourceProperties` method on the resource.

3.5.1.2 Invalid Properties

When a `GetResourceProperty` operation returns a the `InvalidResourcePropertyQName` fault, the property requested is invalid for this property document and the QName of the property does not exist in the property document for the WS-Resource. A resource property of a specific QName is said to exist in a properties document if a resource property element with the same QName appears in the root of the Property document.

Similarly, a client receives `InvalidResourcePropertyQName` in response to a `GetMultipleResourceProperties` request on a property document when one or more of the QNames in the request message are not valid properties in the resource property document root. Note that WS-RF does not mandate the provision of `GetMultipleResourceProperties`: in some implementations, this operation may not be available.

In such a case, the client should issue `GetResourceProperties` in order to establish which of the resource properties are valid.

3.5.1.3 Properties that are valid but do not have a value

If a client issues `GetResourceProperty` on a property with minOccurs='0', the `GetResourceProperty` operation returns a `null` response. This response indicates that there is currently no property instance available for this property, however this is a valid property.

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

WS-RF does not mandate the provision of `GetMultipleResourceProperties`: in some implementations, this operation may not be available.

3.5.1.4 Properties that have the value of 'nil'

If a property is declared nillable and is 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 on the request message. Those properties whose value is nil will be decorated with the xsi:nil="true" attribute.

WS-RF does not mandate the provision of `GetMultipleResourceProperties`: in some implementations, this operation may not be available.

To do: A nice example of this might be the WSRL nil-ing of the termination time property to imply that the resource will not be destroyed for an indefinite period of time
xsd:element name = TerminationTime.Nillible = "true"

returning <wsrl:TerminationTime xsi:nil="true"/>

3.6 Resource Property Attributes

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

3.6.1 Best Practice and Examples

3.6.1.1 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 to which the sale price 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 *3.6.1.2 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>
        </xsdcomplexType>
</xsd:element>
```

Comment: (sgg) Consider adding an example of what an instance resource property document might look like. Also consider adding what a getRP request and response would look like.

3.6.1.2 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>
    </xsdcomplexType>
</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.

3.7 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 GenericDiskDrive resource properties document declaration in [WS-ResourceProperties] might be extended as follows to introduce a mandatory attribute to indicate the document's creation date:

```
<xsd:element name="GenericDiskDriveProperties">
   <xsd:attribute name="CreationDate" type="xsd:DateTime"
use="required"/>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="tns:NumberOfBlocks"/>
            <xsd:element ref="tns:BlockSize"/>
            <xsd:element ref="tns:Manufacturer"/>
            <xsd:any minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="tns:StorageCapability"
```

```
                        minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsdcomplexType>
</xsd:element>
```

Comment: (sgg) Show an instance RP document as example?

In order to query attributes on the root of the Resource Properties document, a WS-Resource client must extract the complete resource property document (`GetResourcePropertyDocument` MEP) and query the specific attribute(s). Similarly, to perform a resource property document root attribute update, the client application must update the attribute in the resource property document and then update the resource property document associated with the resource (using `PutResourcePropertyDocument` MEP).

3.8 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. 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).

3.8.1 Best practice and Examples

3.8.1.1 Resource Property Value-Change Notification Pattern

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

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

Comment: well, it will be..

3.8.1.2 Dynamic Resource Property Value Existence Notification

There may be occasions when an application requires notification that a dynamic Resource Property has come into existence when xsd:any has been specified in the Resource Property document.

> To do: For example, the Printer WS-Resource might have an additional ResourceProperty defined dynamically in order to state the current level of trace. In normal running, this property would not exist, but it would be set during debug. A debug service might be notified automatically when this property comes into existence in order to start displaying the debug?

3.8.1.3 Resource Property 'any' Value Change Notification

This is a WSDM requirement: see WSRF 55.

Resolution was AnyResourcePropertyValueChange in distinguished TopicSpace
I think that this would be a useful example. Based on the primer - perhaps a management function that requires notification of printer property changes?

3.8.1.4 Value-Change Notification Message in WSDM Event Format (Management event)Common Base Events Format

3.8.1.4.1 Extend the Value change notification example to include CBE format. In this case RPVChangeNotification is wrapped inside CBE format. : Bryan has volunteered to help with this.
cf issue wsrf20:
Proposed Recommendations

Comment: Needs to be completed before we can close WSRF 20

Add clarification text about how WS-Topics explains that the notification schema snippet can occur anywhere in the message, allowing different formats to be used for notification messages. The text will need to be explanatory enough to indicate how interoperability can be achieved with different notification formats.

Resolution: the clarification and additional example should be included in the AppNotes

3.8.1.5 How the Web client establishes which notifications are available

If a Ws_resource supports notificationProducer, it must include a list of notifications that it supports.This can be used by the web client in order to establish the notifications available.

4 AppNotes Relating to Base Faults

4.1 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.

4.1.1 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 in both the recipient of the fault as well as the fault generator. This also eases tools development

Faults are generated 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, system fault may be generated as part of transport processing such as a SOAP event). 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.

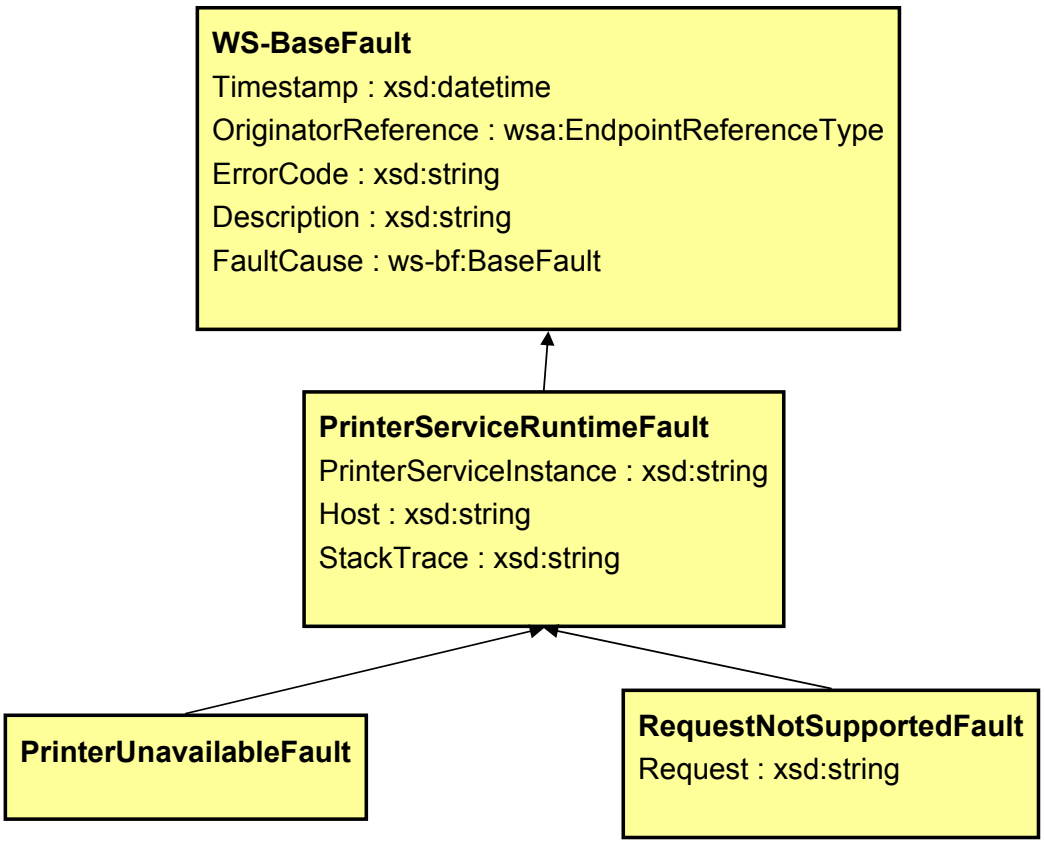
4.1.1.1 Defining and Generating Base Faults

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

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
- and some more...?.

683 Consider the Printer example used previously. We require extended fault types for use by our
684 printer PortType. To begin, it would be prudent to define a *generic* extended fault for the printer
685 service from which all other printer service faults could be derived. This fault type will define the
686 basic information that we require in every fault generated by the printer service. For example,
687 every fault generated should contain an identifier specifying the printer to which the fault was
688 directed, the host, and stack trace detailing the fault cause. Further extended faults may be
689 generated from this underlying generic type (for example: PrinterUnavailableFault and
690 RequestNotSupportedFault).
691

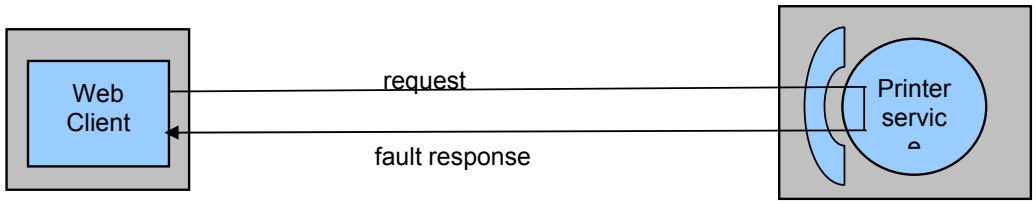


692
693
694 To do: example of what one of these printer faults would look like on the wire.
695
696

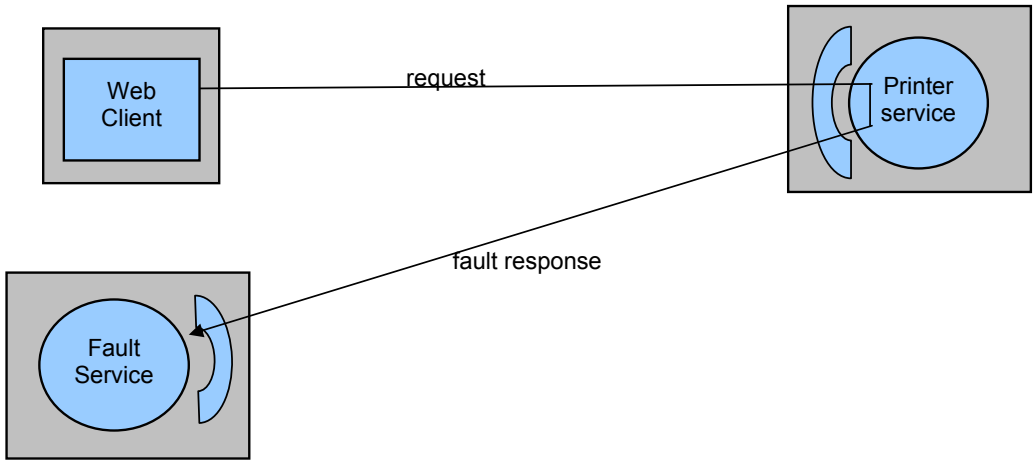
4.1.1.2 Transmitting a Base Fault as a Response Message

Base faults may be returned as a response to an operation on an endpoint or as a one-way notification message. This section describes how a base fault is defined and returned as an operation response - details on how to send a base fault as a one way notification message follow in the next section.

A fault response may be returned synchronously, (for example as part of the HTTP response of the connection that was used for the request flow):



Alternatively, the fault response might be returned asynchronously to the endpoint that initiated the request, or asynchronously to another endpoint specifically dedicated to fault processing:



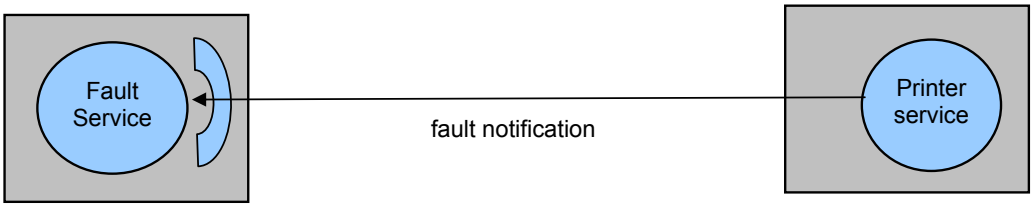
Asynchronous redirection of the fault response at runtime can be achieved by [WS-ADDRESSING].

To do: Illustrate an example operation containing a fault response interface in wsdl

4.1.1.3 Sending a Base Fault as a 1-way Message

This section may be removed if it is not deemed to be adding any useful ws-rf specific information.

Faults may be sent as a one-way notification message. This message may be directed to an endpoint dedicated to fault processing as illustrated below:



To do: Illustrate the operation interface on the receiving endpoint. Describe how the faulting service would send the one way notification.

4.1.1.4 Relaying Base Faults

Faults received by an intermediary should be wrapped by a fault containing the intermediary information and then relayed.

How a fault is relayed by intermediaries nesting and re-transmission. Is this really simply a case of nesting or does the core information of the fault cause require copying to the top level of the fault?

4.1.1.5 Problem Determination from Base Faults

Problem determination of a relayed fault - understanding nested faults eg establishing the originator. This depends on issue 90.

Add the following: Issue 86

- An example of how the UnknownResourceFault might be extended by the application to contain application dependent information - for example the security information for this identifier is invalid.

- A note to the web client to say that resource unknown might not be thrown if the fault is thrown at the system/transport level HTTP 404 may be replaced by a resource access pattern.

5 AppNotes relating to Lifecycle and Resource Access

5.1 Resource Lifecycle Management

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

5.1.1 Best Practice and Examples

5.1.1.1 Resource creation

The WSRF group of specifications 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 EPR as part of its behaviour.

Refer to the Primer for examples of Resource creation.

5.1.1.2 Resource Destruction

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

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

5.2 The Resource Access Pattern

A WS-Resource application may need to export resource references in order for messages to be directed at the resources by clients of the WS-Resource.

5.2.1 Best Practice and Examples

5.2.1.1 Resource Access Pattern Embodiment

A WS-Resource can be identified by its service address URI used in conjunction with some additional information in order to uniquely correlate to the instance behind the WS-Resource service. This is known as the WS-Resource Access Pattern (WS-RAP), as defined in [WS-Resource]..

WS_Resource are referenced, according to the WS-RAP by use of WS-Addressing EndpointReferences (EPRs). The address of the Web service endpoint part of the WS-Resource

is contained in the wsa:Address element information item of the endpoint reference. There are two ways in which the resource identifier may appear:

1)in the contents of the wsa:ReferenceProperty element information item of the endpoint reference (Note, the wsa:ReferenceProperty element information item MUST have at least one child element information item)

or

2) embedded as part of the wsa:Address element information item of the endpoint reference.

The address of the Web service endpoint and the resource identifier of the resource must appear in the message according to binding-specific rules outlined in WS-Addressing. For example, in the SOAP binding defined by WS-Addressing, the Web service endpoint address is contained in the wsa:Address element information item in the endpoint reference and appears in the message as the contents of the wsa:To SOAP header, and each direct child element information item (if any) of the wsa:ReferenceParameters element information item appears in the message as a separate SOAP header.

5.2.1.1.1 Example

The following diagram illustrates an example set of components that comprise a small collection of WS-Resources:

In the example above, there is one Web service that has a URL address of "http://www.example.com/service". This Web service provides access to two resources, identified simply as "R1" and "R2". A reference to the WS-Resource associated with this Web service and the resource identified by "R1" would appear as follows:

```
<wsa:EndpointReference>
   <wsa:Address>http://www.example.com/service</wsa:Address>
   <wsa:ReferenceProperties>
      <tns:SomeDisambiguatorElement>R1</tns:SomeDisambiguatorElement>
   </wsa:ReferenceProperties> ?
…
</wsa:EndpointReference>
```

804 An example GetResourceProperties message, in a SOAP/HTTP binding, would look as follows:

```
805   <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
806               xmlns:wsa=" http://www.w3.org/2005/03/addressing"
807               xmlns:wsrf-rp="http://docs.oasis-open.org/wsrf/rp-1">
808     <S:Header>
809         <wsa:To> http://www.example.com/service </wsa:To>
810         <wsa:Action>
811       http://docs.oasis-open.org/wsrf/rpw-1/GetResourceProperty/GetResourcePropertyRequest
812         </wsa:Action>
813         <tns:SomeDisambiguatorElement
814                   wsa:ReferenceParameter="true">R1</tns:SomeDisambiguatorElement>
815       ...
816     </S:Header>
817     <S:Body>
818        <wsrf-rp:GetResourceProperty …
819       ...
820     </S:Body>
821   </S:Envelope>
```

822

823 It is worth noting that correlation between an EPR and resource might be based on more than
824 one ReferenceParameters and the group of ReferenceParameters used for correlation may vary
825 depend on the operation by which the resource is being accessed. This is a pattern exploited by
826 [BPEL4WS].

827

828 Examples of using [WS-Addressing] in order to realise the Resource Access Pattern are
829 described in [WSRFPrimer].

830 These endpoints might be (but are not necessarily) resources fronted by a WS-Resource service.
831 An EPR can be passed between communicating partners and used in order to target requests.
832 For example, an endpoint issuing an asynchronous request message might pass its EPR as part
833 of the message to indicate the target to which the asynchronous response should be directed.
834 The receiving partner can use the response EPR in order to target its response.

835 The benefit of using EPRs to reference WS-Resource instances is that they provide a standard,
836 protocol independent, mechanism to pass references between communicating partners. The
837 client of a WS-Resource instance represented by an EPR does understand the EPR itself, except
838 that it is an opaque element representing a resource to which requests can be targeted. This is
839 more powerful than using a simple URI as the EPR is an XML representation of an endpoint with
840 additional optional information about how that endpoint should be referenced. The standard
841 nature of the EPR means that the client need not have any proprietary logic in order to target the
842 WS-Resource instance.

6 Interoperability Issues

6.1 Interoperability

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

6.2 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 recognised as such by the intermediary.

Comment: (sgg) FWIW, the WS* template contains some text to recommend BP in this area. We should discuss, but as the text itself is ©, we would need to understand the intent and craft different verbiage.

6.2.1 Best practices and Examples

For cases where the prefix is in an XPath, the XPath can be rewritten such that it does not depend upon namespace prefixes. Alternatively, an application deployer might choose to use a different vendor's intermediary that does not alter the message prefixes.

Comment: (sgg) Can we include an example?

Comment: (sgg) We would need to include an example.

7 Composability

Need resolution to WSRF 103 in here.

8 References

8.1 Normative

[WS-Resource] Web Services Resource1.2
http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-wd-??.pdf
http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-wsdl-wd-??.wsdl
http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-schema-wd-??.xsd

[WS-ResourceProperties] Web Services Resource Properties 1.2
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-wd-??.pdf
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-wsdl-wd-??.wsdl
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-schema-wd-**??**.xsd

[WS-ResourceLifetime] Web Services Resource Lifetime 1.2
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-wd-??.pdf
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-wsdl-wd-**??**.wsdl
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-schema-wd-??.xsd

[WS-BaseFaults] Web Services Base Faults 1.2
http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-wd-??.pdf
http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-wsdl-wd-??.wsdl
http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-schema-wd-**??**.xsd

[WS-ServiceGroup] Web Services Service Group 1.2
http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-spec-wd-??.pdf
http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-wsdl-wd-??.wsdl
http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-schema-wd-**??**.xsd

[WS-BaseNotification]
http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-03.pdf

[WS-Notification]
http://docs.oasis-open.org/committes/dowload.php/6661/WSNpubsub-1-0.pdf

[WS-Topics]
http://docs.oasis-open.org/wsn/2004/06/wsn-WS-Topics-1.2-draft-01.pdf

[WS-Addressing] W3C Member Submission
http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/
W3C member submission, August 10, 2004

899 **[**
900
901

902 8.2 Non-Normative

903 **[WSRFPrimer]** WS-RF Primer,
904 http://docs.oasis-open.org/wsrf/...
905 Working Draft 04, 10 June
906
907 **[OGSPrimer]** Open Grid Service Infrastructure Primer,
908 http://www.ggf.org/oigsi-wg Date?

909

Appendix A. Acknowledgments

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

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), David Martin (IBM), Samuel Meder (Argonne National Laboratory), Jeff Mischkinsky (Oracle), Roger Menday (Forschungszentrum Jlich GmbH), Bryan Murray (Hewlett-Packard), Mark Peel (Novell), Alain Regnier (Ricoh Company, Ltd.), Ian Robinson (IBM), Tom Rutt (Fujitsu), Matsunori 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

[This appendix is optional, but helpful. It should be removed for specifications that are at OASIS Standard level.]

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: - Add updates to do section so discussions aren't forgotten. - moved non-normative txt from WSRP section 4 to appnotes - corrections and minor amendments - improve RAP embodiments section - Remove QOS section
0.6	05/16/2005	Katy Warr	Draft resolutions to the following issues: 22, 63, 52, 89, 95
wd-03	05/20/2005	Ian Robinson	Editorial consistency

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.