



WSRF Application Notes

Public Review Draft 01, 13 June 2005

Document identifier: `wsrf-application_notes-1.2-notes-pr-01`

Location:

http://docs.oasis-open.org/wsrf/wsrf-application_notes-1.2-notes-pr-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 "public review 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/..](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

26			
27	1	Introduction	3
28	2	Who Should Read This Document?	4
29	3	AppNotes relating to Resource Properties	5
30	3.1	Defining a WS-Resource in WSDL 1.1	5
31	3.1.1	Best Practice and Examples	5
32	3.2	Creating a Resource Property document by extending an existing schema	10
33	3.3	Why Resource Property Documents and Resource Properties must be GEDs?	11
34	3.4	Defining a Resource Property that must always exist.....	12
35	3.5	Invalid or Non-existent Resource Properties	13
36	3.5.1	Best Practice and Examples	13
37	3.6	Resource Property Attributes	14
38	3.6.1	Best Practice and Examples	14
39	3.7	Adding Attributes to the Root of the Resource Properties Document.....	15
40	3.8	Notifying Clients of Resource Property changes	16
41	3.8.1	Best practice and Examples	16
42	4	AppNotes Relating to Base Faults	18
43	4.1	Fault Handling	18
44	4.1.1	Best Practice and Examples	18
45	5	AppNotes relating to Lifecycle and Resource Access	22
46	5.1	Resource Lifecycle Management.....	22
47	5.1.1	Best Practice and Examples	22
48	5.2	The Resource Access Pattern	22
49	5.2.1	Best Practice and Examples	22
50	6	Interoperability Issues	25
51	6.1	Interoperability	25
52	6.2	Intermediary changes of Namespace prefixes.....	25
53	6.2.1	Best practices and Examples	25
54	7	Composability	26
55	8	References.....	27
56	8.1	Normative	27
57	8.2	Non-Normative	28
58		Appendix A. Acknowledgments	29
59		Appendix B. Revision History	30
60		Appendix C. Notices	31
61			

62 **1 Introduction**

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

65 The purpose of this document is to answer common questions that might arise during WS-RF
66 application development by means of non-normative scenarios and examples. Additionally, this
67 document may also serve as an aid to clarify potential usage scenarios of WS-RF. The intended
68 audience of this document are WS-Resource application designers

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

72

2 Who Should Read This Document?

73 This document provides a guide for WS-RF application developers who already understand the
74 fundamentals of developing a WS-RF application. The AppNotes is likely to be used as a
75 reference document in order to answer common questions that might arise during application
76 development. For example: *How should multiple WS-RF PortTypes be combined to produce a*
77 *single derived PortType?* Where there is no definitive answer to a question, the appNotes
78 provides a recommended best practice. This document assumes familiarity with the WS-RF
79 specifications and the examples introduced in the [WSRFPrimer].

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

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

85

86

87

88

89 3 AppNotes relating to Resource Properties

90 3.1 Defining a WS-Resource in WSDL 1.1

91 A designer of a WS-Resource application may need to derive a WS-Resource PortType by
92 extending or aggregating one or more existing WS-Resource PortType(s). The newly created
93 PortType will also have associated resource property document that may be derived from the
94 existing PortType(s). There is a trade-off in PortType design between the freedom of the
95 designer to design a PortType and the ability to extend the PortType in future (which may place
96 restrictions on the way that a Resource Properties document is assembled).

97 3.1.1 Best Practice and Examples

98 3.1.1.1 Resource Properties and Interface Aggregation

99 Web service interface designers MAY define a collection of discrete interfaces (portTypes), each
100 of which defines a set of message exchange patterns (operations). A common design scenario is
101 one in which the designer combines these discrete interfaces to form a composed, *most-derived*
102 interface of a Web service. Examples of independently-specified interfaces designed for purposes
103 of aggregation into a most-derived interface include WS-Notification [WS-Notification], WS-
104 ResourceLifetime [WS-ResourceLifetime], and a large number of general-purpose or application-
105 domain-specific management interfaces. Further, there may be various dependencies between
106 these interfaces. That is, the messages defined by interface A may only be useful in a service
107 implementation when combined with those of interface B.

108 Within WSDL 1.1, there is no formally-defined interface extension mechanism¹. In WSDL 1.1 we
109 expect service designers to *copy-and-paste* operations from the various constituent interfaces
110 into a single, flat, most-derived service interface. In addition, we expect the service interface
111 designer to compose a resource property document for the most-derived Web service interface
112 that consists of all of the resource property element declarations from each of the constituent
113 interfaces used in the composition.

114 Consider the following example, wherein a designer extends the "GenericDiskDrive" WS-
115 Resource interface in a vendor-specific fashion.

```
116 <wsdl:definitions ...  
117     xmlns:gen="http://example.com/diskDrive"  
118     xmlns:ven="http://vendor.com/diskDrive"  
119     ...>  
120 ...  
121 <wsdl:types>  
122     <xsd:schema targetNamespace="http://vendor.com/diskDrive" ... >  
123  
124         <!-- Resource property element declarations -->  
125         <xsd:element name="VendorExtension" type="xsd:string" />  
126     </xsd:schema>  
127 </wsdl:types>
```

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

```

127 <!-- Resource properties document declaration -->
128 <xsd:element name="VendorDiskDriveProperties">
129   <xsd:complexType>
130     <xsd:sequence>
131       <xsd:element ref="gen:NumberOfBlocks"/>
132       <xsd:element ref="gen:BlockSize" />
133       <xsd:element ref="gen:Manufacturer" />
134       <xsd:element ref="gen:StorageCapability"
135         minOccurs="0" maxOccurs="unbounded" />
136       <xsd:element ref="ven:VendorExtension" />
137       <xsd:any minOccurs="0" maxOccurs="unbounded" />
138     </xsd:sequence>
139   </xsd:complexType>
140 </xsd:element>
141 ...
142 </xsd:schema>
143 </wsdl:types>
144 ...
145 <!-- Association of resource properties document to a portType -->
146 <wsdl:portType name="VendorDiskDrive"
147   wsrf-rp:ResourceProperties="ven:VendorDiskDriveProperties" >
148   <operation name="...
149 ...
150   <!-- copy/paste operations from genericDiskDrive -->
151   <operation name="start" .../>
152   <operation name="stop" .../>
153 ...
154   <!-- define Vendor-specific operations -->
155   <operation name="reset" .../>
156 ...
157 </wsdl:portType>
158 ...
159 </wsdl:definitions>
160

```

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

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

- 166 1. Define the new portType.
 167 In this example the new portType is named "VendorDiskDrive". This portType extends
 168 "GenericDiskDrive".
- 169 2. Copy all of the operation child elements from the portType being extended, and paste
 170 them as child elements of the new portType; the order of the operations SHOULD be
 171 preserved.
 172 In this example, the "start" and "stop" operations are copied from the GenericDiskDrive
 173 portType and pasted as child elements of the VendorDiskDrive portType.
- 174 3. Define additional, vendor-specific operations as child elements of the new portType.
 175 In this example, the "reset" operation is a new operation defined by the VendorDiskDrive
 176 portType.

- 177 4. Define a new resource properties document, as an XML global element declaration,
178 following the requirements defined in [WS-ResourceProperties].
179 In this example, the element is named "VendorDiskDriveProperties" and defined in the
180 "http://vendor.com/diskDrive" namespace.
- 181 5. Copy all of the child elements (@ref and xsd:any) from the resource properties document
182 of the portType being extended, and paste them as child elements of the new resource
183 properties document; the order of the elements SHOULD be preserved. This step MUST
184 be repeated for each portType that is being extended by this new portType. Any duplicate
185 child elements MUST be removed.
186 In this example, the elements that reference (@ref) "gen:NumberOfBlocks",
187 "gen:Blocksize", and "gen:Manufacturer", "gen:StorageCapability" and the "xsd:any" are
188 copied from the GenericDiskDriveProperties declaration and pasted to the
189 VendorDiskDriveProperties declaration.
- 190 6. Define any additional resource property elements that are specific to the newly-defined
191 resource properties document type.
192 In this example, VendorDiskDriveProperties resource document defines an additional
193 resource property named VendorExtension.

194

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

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

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

201

202

```
203 <xsd:element name="printer-properties">  
204   <xsd:complexType>  
205     <xsd:sequence>  
206       <xsd:element ref="printer-reference"/>  
207       <xsd:element ref="printer-name"/>  
208       <xsd:element ref="printer-state"/>  
209       <xsd:element ref="printer-is-accepting-jobs"/>  
210       <xsd:element ref="queued-job-count"/>  
211       <xsd:element ref="operations-supported"/>  
212       <xsd:element ref="document-format-supported"/>  
213       <xsd:element ref="job-properties" minOccurs="0"  
214 maxOccurs="unbounded"/>  
215     </xsd:sequence>  
216   </xsd:complexType>  
217 </xsd:element>
```

218

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

221

222

```
xmlns:wsrfr-pr=
```

```

223 "http://www.WSRF-
224 Examples.org/IPPprinters/PrinterResourceProperties.xsd"
225
226 :
227
228 <wsdl:portType name="Printer" wsrf-rp:ResourceProperties="wsrf-
229 pr:printer-properties">
230
231 <!--Operations supported by the Print PortType -->
232 <wsdl:operation name="Print-Job"> ... </wsdl:operation>
233 <wsdl:operation name="Create-Job"> ... </wsdl:operation>
234 <wsdl:operation name="Send-URI"> ... </wsdl:operation>
235
236 <!-- WS-RF operations supported by this portType -->
237 <wsdl:operation name="GetResourceProperty"> ... </wsdl:operation>
238 <wsdl:operation name="GetMultipleResourceProperties"> ...
239 </wsdl:operation>
240 <wsdl:operation name="QueryResourceProperties"> ...
241 </wsdl:operation>
242 <wsdl:operation name="SetResourceProperties"> ... </wsdl:operation>
243
244 :
245 </wsdl:portType>

```

246

247 In particular, we require support for the following two operations:

248 Print-URI – submit a document for printing – document identified by the URI.

249 Send-URI - Add to a multi-document job – document identified by the URI.

250 The new portType will be called *URI-Printer* and it extends the more generic *Printer* portType.

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

254

```

255 <wsdl:portType name="URI-Printer"
256
257 <!--Operations supported by the URI-Printer PortType -->
258 <wsdl:operation name="Print-URI"> ... </wsdl:operation>
259 <wsdl:operation name="Send-URI"> ... </wsdl:operation>
260
261
262 <!--Operations from by the Printer PortType -->
263 <wsdl:operation name="Print-Job"> ... </wsdl:operation>
264 <wsdl:operation name="Create-Job"> ... </wsdl:operation>
265
266
267 <!-- WS-RF operations supported by this portType -->
268 <wsdl:operation name="GetResourceProperty"> ... </wsdl:operation>
269 <wsdl:operation name="GetMultipleResourceProperties"> ...
270 </wsdl:operation>
271 <wsdl:operation name="QueryResourceProperties"> ...
272 </wsdl:operation>
273 <wsdl:operation name="SetResourceProperties"> ... </wsdl:operation>
274
275 :
276 </wsdl:portType>

```


277

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

281

```

282 <xsd:element name="uri-printer-properties">
283   <xsd:complexType>
284     <xsd:sequence>
285       <xsd:element ref="printer-reference"/>
286       <xsd:element ref="printer-name"/>
287       <xsd:element ref="printer-state"/>
288       <xsd:element ref="printer-is-accepting-jobs"/>
289       <xsd:element ref="queued-job-count"/>
290       <xsd:element ref="operations-supported"/>
291       <xsd:element ref="document-format-supported"/>
292       <xsd:element ref="job-properties" minOccurs="0"
293 maxOccurs="unbounded"/>
294     </xsd:sequence>
295   </xsd:complexType>
296 </xsd:element>

```

297

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

300

```

301 xmlns:wsrf-upr="http://???/URIPrinterResourceProperties.xsd"
302
303
304 <wsdl:portType name="URI-Printer" wsrf-rp:ResourceProperties="wsrf-
305 upr:uri-printer-properties">
306   :
307 </wsdl:portType>

```

308

309 3.1.1.2 Creating a new Port Type by Aggregating Existing PortTypes

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

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

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

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

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

323

324 3.1.1.3 Establishing Aggregated PortType Derivation

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

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

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

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

337

338 3.1.1.4 Creating a new PortType by Aggregating Existing PortTypes which 339 have a Resource Property with the same Name but different 340 Cardinality

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

344 This scenario presents no problem as long as the semantics (in particular, the cardinality) of the
345 shared property are the same in both cases. However, if the semantics of this property differ
346 between the existing PortTypes, there may be ambiguity regarding the derived property's
347 behaviour. For example, the property is defined as mandatory in one of the derived
348 from PortTypes, and optional in the other.

349 The best solution to this problem is to re-visit the design of the derived-from PortTypes and
350 consider why it is that the same property occurs in each with different cardinality. The PortTypes
351 should be restructured to remove the cardinality conflict.

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

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

359 3.2 Creating a Resource Property document by extending an 360 existing schema

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

363

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

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)?



```

366     <xsd:sequence>
367         <xsd:element ref="tns:NumberOfBlocks"/>
368         <xsd:element ref="tns:BlockSize"/>
369         <xsd:element ref="tns:Manufacturer"/>
370         <xsd:any minOccurs="0" maxOccurs="unbounded"/>
371         <xsd:element ref="tns:StorageCapability"
372             minOccurs="0" maxOccurs="unbounded"/>
373     </xsd:sequence>
374 </xsdcomplexType>
375 </xsd:element>

```

376

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

```

378 <xsd:element name="GenericDiskDrivePropertiesWithCreationDate">
379     <xsd:complexType>
380         <xsd:complexContent>
381             <xsd:extension base="tns:GenericDiskDriveProperties">
382                 <xsd:attribute name="CreationDate"
383                     type="xsd:DateTime" use="required"/>
384             </xsd:extension>
385         </xsd:complexContent>
386     </xsd:complexType>
387 </xsd:element>

```

388

389

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

390 3.3 Why Resource Property Documents and Resource Properties 391 must be GEDs?

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

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

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

404

```

405 <xsd:element name="InvalidGenericDiskDriveProperties">
406     <xsd:complexType>
407         <xsd:sequence>
408             <xsd:element name="tns:Blocks">
409                 <xsd:complexType>
410                     <xsd:sequence>
411                         <xsd:element ref="tns:Number"/>
412                         <xsd:element ref="tns:Size"/>

```

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.



```

413         </xsd:sequence>
414     </xsd:complexType>
415 </xsd:element>
416     <xsd:element ref="tns:NumberOfBlocks"/>
417     <xsd:element ref="tns:Size"/>
418     <xsd:element ref="tns:Manufacturer"/>
419     <xsd:any minOccurs="0" maxOccurs="unbounded"/>
420     <xsd:element ref="tns:StorageCapability"
421         minOccurs="0" maxOccurs="unbounded"/>
422 </xsd:sequence>
423 </xsd:complexType>
424 </xsd:element>

```

425

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

429

430 3.4 Defining a Resource Property that must always exist

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

434

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

439

```

440 <xsd:element name="GenericDiskDriveProperties">
441     <xsd:complexType>
442         <xsd:sequence>
443             <xsd:element ref="tns:NumberOfBlocks"/>
444             <xsd:element ref="tns:BlockSize"/>
445             <xsd:element ref="tns:Manufacturer"
446                 minOccurs="1" maxOccurs="1"/>
447             <xsd:any minOccurs="0" maxOccurs="unbounded"/>
448             <xsd:element ref="tns:StorageCapability"
449                 minOccurs="0" maxOccurs="unbounded"/>
450         </xsd:sequence>
451     </xsd:complexType>
452 </xsd:element>

```

453

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

458

459 **3.5 Invalid or Non-existent Resource Properties**

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

462 **3.5.1 Best Practice and Examples**

463 **3.5.1.1 Establishing a list of Valid Resource Properties**

464

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

470

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

473 **3.5.1.2 Invalid Properties**

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

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

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

486

487 **3.5.1.3 Properties that are valid but do not have a value**

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

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

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

497 3.5.1.4 Properties that have the value of 'nil'

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

500 Similarly, the `GetMultipleResourceProperties` operation returns a collection of the
501 properties corresponding to the QNames on the request message. Those properties whose
502 value is nil will be decorated with the `xsi:nil="true"` attribute.

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

505 To do: A nice example of this might be the WSRL nil-ing of the termination time
506 property to imply that the resource will not be destroyed for an indefinite period of
507 time

508 `xsd:element name = TerminationTime.Nillible = "true"`

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

510

511 3.6 Resource Property Attributes

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

516 3.6.1 Best Practice and Examples

517 3.6.1.1 Refining a Resource Property Definition with Lifetime Attributes

518 Consider a WS-Resource that represents an item in a warehouse. The item may have a price
519 associated (by means of a resource property) and, on occasion, a sale price. In this example, the
520 resource property defining the sale price is only valid between specific dates so it is necessary to
521 indicate to the consumer some lifetime aspects of this property. In this case `GoodFrom` and
522 `GoodUntil` are attributes added to the sale price resource property to indicate the date that the
523 sale price came into effect and the date to which the sale price is valid. Attributes such as
524 `GoodFrom` and `GoodUntil` can appear on a resource property if

- 525 • the Resource Property definition explicitly includes them as attributes

526 or if,

- 527 • the Resource Property definition allows attribute extensibility by associating `anyAttribute`
528 with the property definition (as described in the example in section 3.6.1.2 *Providing Attribute*
529 *Extensibility to a Resource Property*).

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

532

```
533 <xsd:element name="SalePrice">  
534   <xsd:complexType>  
535     <xsd:simpleContent>  
536       <xsd:extension base="xsd:decimal">  
537         <xsd:attribute name="GoodFrom" type="xsd:dateTime"/>
```

538
539
540
541
542

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

543

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.

544 3.6.1.2 Providing Attribute Extensibility to a Resource Property

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

548 The xml schema definition for the `currentTime` resource property element type is as follows:

549

550
551
552
553
554
555
556
557
558
559

```
<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>
```

560

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

564 3.7 Adding Attributes to the Root of the Resource Properties 565 Document

566 No restrictions are placed on the adding of attributes to the Resource Properties document root.
567 Consumers of the WS-Resource can retrieve attributes on the Resource Properties Document
568 root by exploiting the `GetResourcePropertyDocument` MEP.

569 For example, the `GenericDiskDrive` resource properties document declaration in [WS-
570 ResourceProperties] might be extended as follows to introduce a mandatory attribute to indicate
571 the document's creation date:

572

573
574
575
576
577
578
579
580
581
582

```
<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"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

583
584
585
586

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

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

587

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

594

595

596

597 **3.8 Notifying Clients of Resource Property changes**

598 A common pattern for WS-Resource applications is for the WS-Resource to send automatic
599 notifications of changes to resource property elements of its resources to interested partners.
600 Resource properties may be changed by external events (for example, by use of
601 `setResourceProperties`) or by events internal to the service and not directly visible to the
602 client (for example, a printer is put out of service).

603 **3.8.1 Best practice and Examples**

604 **3.8.1.1 Resource Property Value-Change Notification Pattern**

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

607

608

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

Comment: well, it will be..

611

612 **3.8.1.2 Dynamic Resource Property Value Existence Notification**

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

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

621



622 **3.8.1.3 Resource Property 'any' Value Change Notification**

623 This is a WSDM requirement: see WSRF 55.
624 Resolution was AnyResourcePropertyValueChange in distinguished TopicSpace
625 I think that this would be a useful example. Based on the primer - perhaps a
626 management function that requires notification of printer property changes?

627 **3.8.1.4 Value-Change Notification Message in WSDM Event Format**
628 **(Management event)Common Base Events Format**

629 **3.8.1.4.1** Extend the Value change notification example to include CBE format. In this
630 case RPVChangeNotification is wrapped inside CBE format. : Bryan has
631 volunteered to help with this.
632 cf issue wsr20:

633 **Proposed Recommendations**

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

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

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

640 **3.8.1.5 How the Web client establishes which notifications are available**

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

644



645 4 AppNotes Relating to Base Faults

646 4.1 Fault Handling

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

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

652 4.1.1 Best Practice and Examples

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

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

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

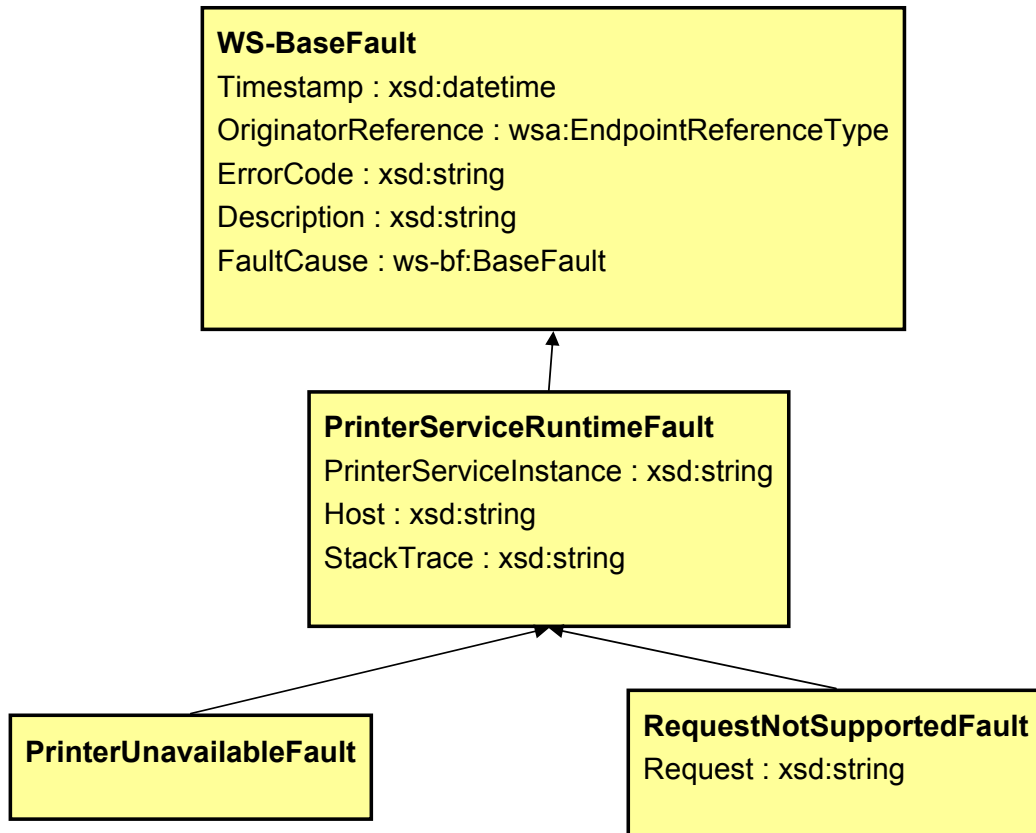
668 4.1.1.1 Defining and Generating Base Faults

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

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

- 676 • Host: The host on which the fault was generated. If the service could be run on more than
677 one host (for example, in the case of load balancing), it is important to include the host name
678 as an element in the extended fault type.
- 679 • Process: The process in which the service was running when the fault occurred.
- 680 • SOAPFault information: If the WS-BaseFault is to wrap a soap fault then the soap fault code
681 and role should be contained in the fault
- 682 • 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
695
696

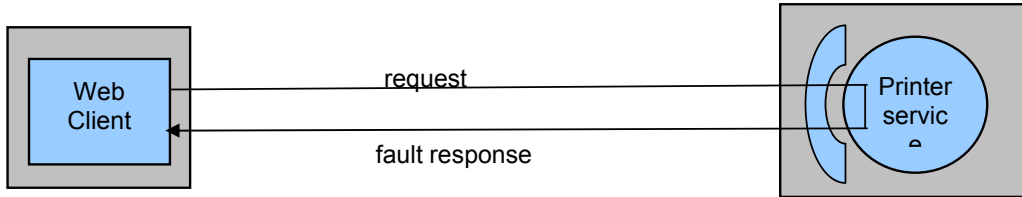
To do: example of what one of these printer faults would look like on the wire.

697 **4.1.1.2 Transmitting a Base Fault as a Response Message**

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

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

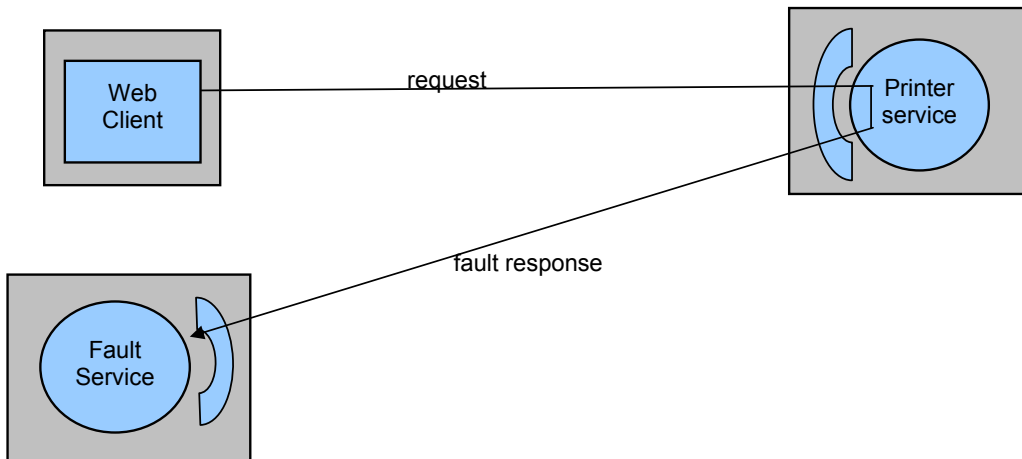
704



705

706

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



709

710

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

713

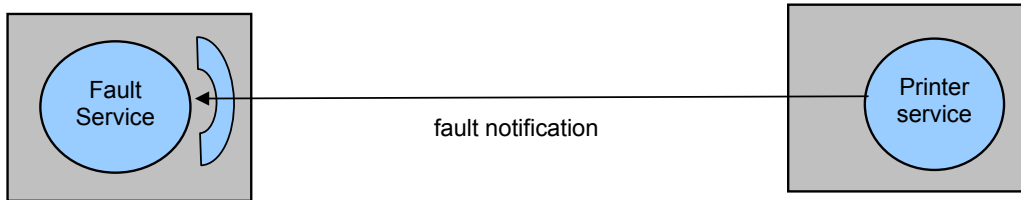
714 To do: Illustrate an example operation containing a fault response interface in
715 wsdl

716

717 **4.1.1.3 Sending a Base Fault as a 1-way Message**

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

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



722

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

725 **4.1.1.4 Relaying Base Faults**

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

728

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

732 **4.1.1.5 Problem Determination from Base Faults**

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

735 Add the following: Issue 86

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

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

741

742 **5 AppNotes relating to Lifecycle and Resource**
743 **Access**

744 **5.1 Resource Lifecycle Management**

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

747 **5.1.1 Best Practice and Examples**

748 **5.1.1.1 Resource creation**

749 The WSRF group of specifications do not specify how a WS-Resource instance should be
750 created. A commonly used pattern is the Factory Pattern whereby a separate Web service
751 (factory) exposes an operation to clients for creating a new resource instance and returning a
752 reference to that instance.

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

757 Refer to the Primer for examples of Resource creation.

758 **5.1.1.2 Resource Destruction**

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

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

764 **5.2 The Resource Access Pattern**

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

767 **5.2.1 Best Practice and Examples**

768 **5.2.1.1 Resource Access Pattern Embodiment**

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

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

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

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

780 or

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

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

789 5.2.1.1.1 Example

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



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

```
797 <wsa:EndpointReference>
798   <wsa:Address>http://www.example.com/service</wsa:Address>
799   <wsa:ReferenceProperties>
800     <tns:SomeDisambiguatorElement>R1</tns:SomeDisambiguatorElement>
801   </wsa:ReferenceProperties> ?
802   ...
803 </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:wsrp="http://docs.oasis-open.org/wsrp/1">
808   <S:Header>
809     <wsa:To> http://www.example.com/service </wsa:To>
810     <wsa:Action>
811       http://docs.oasis-open.org/wsrp/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.

843 **6 Interoperability Issues**

844 **6.1 Interoperability**

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

847 **6.2 Intermediary changes of Namespace prefixes**

848 SOAP allows intermediaries to modify the XML namespace prefixes of messages passing
849 through them. This may result in interoperability problems if attributes or text nodes in the
850 message contain `QNames`, as the `QName` prefixes in attributes and text nodes will not recognised
851 as such by the intermediary.

852 **6.2.1 Best practices and Examples**

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

856

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.

Comment: (sgg) Can we include an example?

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



857 **7 Composability**

858 Need resolution to WSRF 103 in here.

859

8 References

860

8.1 Normative

861

[WS-Resource] Web Services Resource 1.2

862

http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-wd-?.pdf

863

http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-wsdl-wd-?.wsdl

864

http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-schema-wd-?.xsd

865

866

[WS-ResourceProperties] Web Services Resource Properties 1.2

867

http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-wd-?.pdf

868

http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-wsdl-wd-?.wsdl

869

http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-schema-wd-?.xsd

870

871

[WS-ResourceLifetime] Web Services Resource Lifetime 1.2

872

http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-wd-?.pdf

873

http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-wsdl-wd-?.wsdl

874

http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-schema-wd-?.xsd

875

876

[WS-BaseFaults] Web Services Base Faults 1.2

877

http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-wd-?.pdf

878

http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-wsdl-wd-?.wsdl

879

http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-schema-wd-?.xsd

880

881

[WS-ServiceGroup] Web Services Service Group 1.2

882

http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-spec-wd-?.pdf

883

http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-wsdl-wd-?.wsdl

884

http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-schema-wd-?.xsd

885

886

[WS-BaseNotification]

887

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

888

889

[WS-Notification]

890

<http://docs.oasis-open.org/committes/download.php/6661/WSNpubsub-1-0.pdf>

891

892

[WS-Topics]

893

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

894

895

[WS-Addressing] W3C Member Submission

896

<http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/>

897

W3C member submission, August 10, 2004

898

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

910 Appendix A. Acknowledgments

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

913

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

930

931

Appendix B. Revision History

932
933

[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: <ul style="list-style-type: none"> - 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
pr-01	06/13/2005	Ian Robinson	Change status to PR draft

934

935

Appendix C. Notices

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

945

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

949

950 Copyright (C) OASIS Open (2005). All Rights Reserved.

951

952 This document and translations of it may be copied and furnished to others, and derivative works
953 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
954 published and distributed, in whole or in part, without restriction of any kind, provided that the
955 above copyright notice and this paragraph are included on all such copies and derivative works.
956 However, this document itself may not be modified in any way, such as by removing the copyright
957 notice or references to OASIS, except as needed for the purpose of developing OASIS
958 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
959 Property Rights document must be followed, or as required to translate it into languages other
960 than English.

961

962 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
963 successors or assigns.

964

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