

# 1 **Web Services Resource Transfer (WS-RT)**

2 **Version 1.0, August 2006**

## 3 **Authors**

4 Brian Reistad, Microsoft Corporation

5 Bryan Murray, HP

6 Doug Davis, IBM

7 Ian Robinson (Editor), IBM

8 Raymond McCollum (Editor), Microsoft Corporation

9 Alexander Nosov, Microsoft Corporation

10 Steve Graham, IBM

11 Vijay Tewari, Intel Corporation

12 William Vambenepe, HP

## 13 **Copyright Notice**

14 (c) 2006 Hewlett-Packard Development Company (HP), Intel Corporation,  
15 International Business Machines Corporation (IBM), and Microsoft Corporation. All  
16 rights reserved.

17 Permission to copy and display the "Web Services Resource Transfer" Specification,  
18 in any medium without fee or royalty is hereby granted, provided that you include  
19 the following on ALL copies of the "Web Services Resource Transfer" Specification, or  
20 portions thereof, that you make:

21 1. A link or URL to the "Web Services Resource Transfer" Specification at this  
22 location: <http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer>.

23 2. The copyright notice as shown in the "Web Services Resource Transfer"  
24 Specification.

25 Hewlett-Packard Development Company (HP), Intel Corporation, International  
26 Business Machines Corporation (IBM), and Microsoft Corporation (collectively, the  
27 "Authors") each agree to grant you a royalty-free license, under reasonable, non-  
28 discriminatory terms and conditions to their respective patents that they deem  
29 necessary to implement the "Web Services Resource Transfer" Specification.

30 THE "WEB SERVICES RESOURCE TRANSFER" SPECIFICATION IS PROVIDED "AS IS,"  
31 AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR  
32 IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,  
33 FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE  
34 CONTENTS OF THE "WEB SERVICES RESOURCE TRANSFER" SPECIFICATION ARE  
35 SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH

36 CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS,  
 37 TRADEMARKS OR OTHER RIGHTS.

38 THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,  
 39 INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY  
 40 USE OR DISTRIBUTION OF THE "WEB SERVICES RESOURCE TRANSFER"  
 41 SPECIFICATION.

42 The name and trademarks of the Authors may NOT be used in any manner, including  
 43 advertising or publicity pertaining to the "Web Services Resource Transfer"  
 44 Specification or its contents without specific, written prior permission. Title to  
 45 copyright in the "Web Services Resource Transfer" Specification will at all times  
 46 remain with the Authors.

47 No other rights are granted by implication, estoppel or otherwise.

48 **Abstract**

49 This specification defines extensions to **[WS-Transfer]**. While its initial design  
 50 focuses on management resource access its use is not necessarily limited to those  
 51 situations.

52 **Composable Architecture**

53 The Web service specifications (WS-\*) are designed to be composed with each other  
 54 to provide a rich set of tools for the Web services environment. This specification  
 55 relies on other Web service specifications to provide secure, reliable, and/or  
 56 transacted message delivery and to express Web service metadata.

57 **Status**

58 This specification is an initial draft. It is likely to change and there is no guarantee of  
 59 compatibility between this version and subsequent versions. As a result, it should  
 60 only be used for information, feedback and experimentation.

61

62 **Table of Contents**

63 **1. Introduction** ..... **3**

64    1.1 Requirements ..... 3

65    1.2 Non-Requirements ..... 4

66    1.3 Example ..... 4

67 **2. Terminology and Notation** ..... **7**

68    2.1 Terminology ..... 7

69    2.2 XML Namespaces ..... 7

70    2.3 Notational Conventions ..... 8

71    2.4 Compliance ..... 9

72 **3. Extensions to WS-Transfer** ..... **9**

73    3.1 Fragments ..... 9

74    3.2 Expression Dialect ..... 9

75	3.3 Get.....	12
76	3.4 Put .....	16
77	3.5 Create.....	21
78	<b>4. Faults .....</b>	<b>25</b>
79	4.1 wsa:DestinationUnreachable.....	26
80	4.2 wsa:EndpointUnavailable .....	26
81	4.3 ConcurrencyFault .....	26
82	4.4 UnsupportedDialectFault .....	27
83	4.5 InvalidExpressionFault.....	27
84	4.6 GetFault .....	27
85	4.7 ResourceValidityFault .....	28
86	4.8 FragmentAlreadyExistsFault .....	28
87	4.9 PutFault.....	28
88	4.10 PutModeUnsupportedFault .....	29
89	4.11 CreateFault .....	29
90	4.12 InvalidMetadataFault .....	29
91	4.13 MultipartLimitExceededFault .....	29
92	4.14 InvalidPutSyntaxFault.....	30
93	<b>5. Security.....</b>	<b>30</b>
94	<b>6. Acknowledgements .....</b>	<b>30</b>
95	<b>7. References .....</b>	<b>30</b>
96	<b>Appendix I – XPath Level 1 .....</b>	<b>31</b>
97	<b>Appendix II – Resource Metadata Content .....</b>	<b>34</b>
98	II.A Lifecycle metadata .....	34
99	II.B Expression Dialect metadata.....	35
100	<b>Appendix III – XML Schema.....</b>	<b>36</b>
101	<b>Appendix IV – WSDL .....</b>	<b>43</b>
102		

## 103 1. Introduction

104 This specification is intended to form an essential core component of a unified  
105 resource access protocol for the Web services space.

106 The operations described in this specification constitute an extension to the WS-  
107 Transfer specification, which defines standard messages for controlling resources  
108 using the familiar paradigms of "get", "put", "create", and "delete". The extensions  
109 deal primarily with fragment-based access to resources to satisfy the common  
110 requirements of WS-ResourceFramework and WS-Management.

111 This document constitutes WS-ResourceTransfer, hereafter referred to as WS-RT.

### 112 1.1 Requirements

113 This specification intends to meet the following requirements:

- 114 • Define a standardized technique for accessing resources using semantics  
115 familiar to those in the system management domain: get, put, create and  
116 delete.

- 117 • Define WSDL 1.1 portTypes, for the Web service methods described in this
- 118 specification, compliant with WS-I Basic Profile 1.1 [**WS-I BP 1.1**].
- 119 • Define minimum requirements for compliance without constraining richer
- 120 implementations.
- 121 • Compose with other Web service specifications for secure, reliable, transacted
- 122 message delivery.
- 123 • Provide extensibility for more sophisticated and/or currently unanticipated
- 124 scenarios.
- 125 • Support a variety of encoding formats including (but not limited to) both
- 126 SOAP 1.1 [[SOAP 1.1](#)] and SOAP 1.2 [[SOAP 1.2](#)] Envelopes.

## 127 1.2 Non-Requirements

128 This specification does not intend to meet the following requirements:

- 129 • Discovery of resources

## 130 1.3 Example

131 This section contains a complete example of a WS-RT "Get" operation. This example

132 is meant for illustration only and does not represent normative behavior or content.

133 Table 1 shows the XML representation of the example resource which will be

134 accessed by the protocol operation. The example resource is a physical disk which

135 has a number of logical volumes.

136 **Table 1: Example resource**

```

137 (01) <Disk xmlns="http://example.org/sample">
138 (02)   <DiskCapacity>6250000000</DiskCapacity>
139 (03)   <DiskFreeSpace>524182841</DiskFreeSpace>
140 (04)   <SerialNumber>123-F2560</SerialNumber>
141 (05)   <LastAuditDate>1998-05-25T13:30:15</LastAuditDate>
142 (06)   <Volume>
143 (07)     <Drive>C:</Drive>
144 (08)     <Label>MyDrive-C</Label>
145 (09)     <TotalCapacity>10000000000</TotalCapacity>
146 (10)     <FreeSpace>6234794528</FreeSpace>
147 (11)   </Volume>
148 (12)   <Volume>
149 (13)     <Drive>D:</Drive>
150 (14)     <Label>MyDrive-D</Label>
151 (15)     <TotalCapacity>30000000000</TotalCapacity>
152 (16)     <FreeSpace>26462809800</FreeSpace>
153 (17)   </Volume>
154 (18)   <Volume>
155 (19)     <Drive>E:</Drive>
156 (20)     <Label>MyDrive-E</Label>
157 (21)     <TotalCapacity>22500000000</TotalCapacity>
158 (22)     <FreeSpace>16056784170</FreeSpace>
159 (23)   </Volume>
160 (24) </Disk>

```

161

162 The protocol message for retrieving parts of the above resource representation is  
163 shown in Table 2. The response message of a WS-Transfer "Get" request message  
164 would return the entire representation of the resource, so this example illustrates a  
165 WS-RT "Get" request message augmented for extracting specific fragments of the  
166 representation:

167

168 **Table 2: Example "Get" operation of resource content**

```
169 (01) <s:Envelope
170 (02)   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
171 (03)   xmlns:wsa="http://www.w3.org/2005/08/addressing"
172 (04)   xmlns:wsrt=
173 (05)     "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer">
174 (06) <s:Header>
175 (07)   <wsa:To>http://www.example.org/disk</wsa:To>
176 (08)   <wsa:Action s:mustUnderstand="true">
177 (09)     http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
178 (10)   </wsa:Action>
179 (11)   <wsrt:ResourceTransfer s:mustUnderstand="true"/>
180 (12)   ...
181 (13) </s:Header>
182 (14) <s:Body>
183 (15)   <wsrt:Get
184 (16)     Dialect="http://schemas.xmlsoap.org/ws/2006/08/
185 (17)       resourceTransfer/Dialect/XPath-Level-1"
186 (18)     xmlns:d="http://example.org/sample">
187 (19)     <wsrt:Expression>
188 (20)       d:Volume[1]/d:Label
189 (21)     </wsrt:Expression>
190 (22)     <wsrt:Expression>
191 (23)       d:DiskCapacity
192 (24)     </wsrt:Expression>
193 (25)     <wsrt:Expression>
194 (26)       d:SerialNumber/text()
195 (27)     </wsrt:Expression>
196 (28)   </wsrt:Get>
197 (29) </s:Body>
198 (30) </s:Envelope>
```

199

200

201 In this example, the operation is a WS-Transfer "Get" as defined by the wsa:Action  
202 in line (09). The extended, fragment-aware Get behavior is indicated by the  
203 wsrt:ResourceTransfer header at line (11). The resource being accessed is  
204 referenced by the WS-Addressing endpoint reference, implied by the wsa:To element  
205 on line (07). WS-RT extensions for extracting fragments of the resource  
206 representation are in the wsrt:Get block on lines (15) through (28). For each

207 targeted fragment the value to be selected is specified in the wsrt:Expression  
208 element.

209

210 The response to the "Get" message is illustrated in Table 3.

211 **Table 3: Example "Get" response.**

```
212 (01) <s:Envelope  
213 (02)   xmlns:s="http://www.w3.org/2003/05/soap-envelope"  
214 (03)   xmlns:wsa="http://www.w3.org/2005/08/addressing"  
215 (04)   xmlns:wsrt=  
216 (05)     "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer">  
217 (06) <s:Header>  
218 (07)   <wsa:To>  
219 (08)   http://www.w3.org/2005/08/addressing/anonymous  
220 (09)   </wsa:To>  
221 (10)   <wsa:Action s:mustUnderstand="true">  
222 (11)   http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse  
223 (12)   </wsa:Action>  
224 (13)   <wsrt:ResourceTransfer/>  
225 (14)   ...  
226 (15) </s:Header>  
227 (16) <s:Body>  
228 (17)   <wsrt:GetResponse xmlns:d="http://example.org/sample">  
229 (18)     <wsrt:Result>  
230 (19)       <d:Label>MyDrive-C</d:Label>  
231 (20)     </wsrt:Result>  
232 (21)     <wsrt:Result>  
233 (22)       <d:DiskCapacity>62500000000</d:DiskCapacity>  
234 (23)     </wsrt:Result>  
235 (24)     <wsrt:Result>  
236 (25)       <wsrt:TextNode>123-F2560</wsrt:TextNode>  
237 (26)     </wsrt:Result>  
238 (27)   </wsrt:GetResponse>  
239 (28) </s:Body>  
240 (29) </s:Envelope>
```

241 Note that the value of each resource fragment requested via a wsrt:Expression  
242 element is individually returned in a matching wsrt:Result element. This example  
243 uses the XPath Level 1 Dialect for the wsrt:Expression elements, one of which shows  
244 the use of the XPath text() NodeTest. The wsrt:Result for the third fragment contains  
245 only the serialized text value of that element (line (25)), rather than the XML  
246 element wrapper as in the other cases.

## 247 2. Terminology and Notation

### 248 2.1 Terminology

249 Some of the terminology defined in this specification is repeated from the WS-  
250 Transfer specification for convenience and is not meant to deviate from those  
251 definitions in any way.

252 Resource

253 A Web service that is addressable by an endpoint reference as defined in WS-  
254 Addressing and that can be represented by an XML document. This  
255 representation can be accessed using the operations defined in the WS-Transfer  
256 and WS-ResourceTransfer specifications.

257 Resource representation

258 The XML representation of the resource that is accessed using the operations  
259 defined in the WS-Transfer and WS-ResourceTransfer specifications.

260 Resource factory

261 A Web service that is capable of creating new resources using the Create  
262 operation defined in WS-Transfer and the WS-ResourceTransfer specifications.

263 Metadata resource

264 A resource whose XML representation describes some aspect of the metadata of  
265 another resource, such as its WSDL or lifecycle metadata. Each resource may  
266 have zero or more metadata resources associated with it.

267 Fragment

268 The term "fragment" is used in this specification to mean a part of the resource  
269 representation.

270 EPR

271 The `wsa:EndpointReference` (EPR), as defined by WS-Addressing, is a reference  
272 to the resource in its entirety. Operations, which are otherwise unconstrained  
273 within this specification, are assumed to affect the resource as a whole.

### 274 2.2 XML Namespaces

275 The XML Namespace URI that MUST be used by implementations of this specification  
276 is:

277 `http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer`

278 Table 4 lists XML namespaces that are used in this specification. The choice of any  
279 namespace prefix is arbitrary and not semantically significant.

280 **Table 4: Prefixes and XML Namespaces used in this specification.**

Prefix	XML Namespace	Specification(s)
s	(Either SOAP 1.1 or 1.2)	(Either SOAP 1.1 or 1.2)
s11	<code>http://schemas.xmlsoap.org/soap/envelope/</code>	[ <a href="#">SOAP 1.1</a> ]
s12	<code>http://www.w3.org/2003/05/soap-envelope</code>	[ <a href="#">SOAP 1.2</a> ]
wsa	<code>http://www.w3.org/2005/08/addressing</code>	[ <a href="#">WS-Addressing</a> ]
wsmex	<code>http://schemas.xmlsoap.org/ws/2004/09/mex</code>	[ <a href="#">WS-MetadataExchange</a> ]
wsdl	<code>http://schemas.xmlsoap.org/wsdl/</code>	[ <a href="#">WSDL 1.1</a> ]

wsrt	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer	<a href="#">[WS-ResourceTransfer]</a>
wxf	http://schemas.xmlsoap.org/ws/2004/09/transfer	<a href="#">[WS-Transfer]</a>
xs	http://www.w3.org/2001/XMLSchema	<a href="#">[XML Schema]</a>

## 281 2.3 Notational Conventions

282 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",  
 283 "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this  
 284 document are to be interpreted as described in RFC 2119 [\[RFC 2119\]](#).

285 This specification uses the following syntax to define outlines for messages:

- 286 • The syntax appears as an XML instance, but values in italics indicate data  
 287 types instead of literal values.
- 288 • Characters are appended to elements and attributes to indicate cardinality:  
 289 ○ "?" (0 or 1)  
 290 ○ "\*" (0 or more)  
 291 ○ "+" (1 or more)
- 292 • The character "|" is used to indicate a choice between alternatives.
- 293 • The characters "(" and ")" are used to indicate that contained items are to be  
 294 treated as a group with respect to cardinality or choice.
- 295 • The characters "[" and "]" are used to call out references and property names.
- 296 • Ellipses (i.e., "...") indicate points of extensibility. Additional children and/or  
 297 attributes MAY be added at the indicated extension points but MUST NOT  
 298 contradict the semantics of the parent and/or owner, respectively. By default,  
 299 if a receiver does not recognize an extension, the receiver SHOULD ignore the  
 300 extension; exceptions to this processing rule, if any, are clearly indicated  
 301 below.
- 302 • XML namespace prefixes (see Table 4) are used to indicate the namespace of  
 303 the element being defined.

304 In addition to Message Information Header properties [\[WS-Addressing\]](#), this  
 305 specification uses the following properties to define messages:

### 306 **[Headers]**

307 Unordered message headers.

### 308 **[Action]**

309 The value to be used for the wsa:Action URI.

### 310 **[Body]**

311 A message body.

312 These properties bind to a SOAP Envelope as follows:

```

313 <s:Envelope>
314   <s:Header>
315     [Headers]
316     <wsa:Action>[Action]</wsa:Action>
317     ...
318   </s:Header>
319   <s:Body>[Body]</s:Body>
320 </s:Envelope>

```



321 This specification defines Fault properties for each defined fault and defines SOAP  
322 bindings for each Fault property.

## 323 **2.4 Compliance**

324 An endpoint is not compliant with this specification if it fails to satisfy one or more of  
325 the MUST or REQUIRED level requirements defined herein.

326 Normative text within this specification takes precedence over the XML Schema and  
327 WSDL descriptions, which in turn take precedence over outlines, which in turn take  
328 precedence over examples.

## 329 **3. Extensions to WS-Transfer**

330 WS-Transfer defines operations to Get, Put, Create and Delete representations of  
331 resources. WS-ResourceTransfer extends these operations to add the capability to  
332 operate on fragments of the resource representations.

### 333 **3.1 Fragments**

334 Since an EPR refers to a resource as a whole, techniques which are used to reference  
335 or access parts of the resource representation are referred to as "fragment access" in  
336 that they access fragments of the XML representing the resource.

337 This specification defines an extensible mechanism for designating the expression  
338 syntax by which the fragment is identified or computed, and defines several such  
339 standard expression syntaxes or "dialects".

340

### 341 **3.2 Expression Dialect**

342 The dialects defined below are used to form an expression that can be evaluated with  
343 respect to the XML document that represents the resource. The de-referenced value  
344 of the expression is the part of the XML that is of interest. The expression may form  
345 a logical "pointer" to the fragment of XML that is of interest or, depending on the  
346 dialect, may form a query that can be applied to the XML document to produce an  
347 evaluated result. It is important to understand that these expression dialects simply  
348 identify the appropriate fragment of the resource representation and that the  
349 **[Action]** itself defines what will happen to the referenced fragment.

350

351 The definition of each dialect must clearly specify how the result of evaluating an  
352 expression against a resource representation is serialized to XML and should specify  
353 any dialect-specific behavior for operations that access the resource representation.

354

#### 355 **3.2.1 QName Dialect**

356 The QName expression dialect is a simple dialect for expressions that uses a QName  
357 to reference the immediate children of the root element of the resource  
358 representation. Consider the resource described in Table 1.

359 In this example, the QName dialect can define references to the elements  
360 <DiskCapacity>, <DiskFreeSpace>, <SerialNumber>, <LastAuditDate> and all  
361 <Volume> elements. The QName dialect cannot define direct references to the  
362 elements <Drive>, <Label>, <TotalCapacity> and <FreeSpace> - since they are not  
363 direct children of the Disk (root) element of the resource - or to *specific* <Volume>  
364 elements. Table 5, below, shows an example usage of this dialect. This dialect is  
365 useful for simple resources with no XPath processing capability.

366 The QName dialect MUST be indicated by using the URI:  
367 `http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/Dialect/QName`  
368 Note that the expression MUST evaluate to zero or more entire elements, each  
369 including the element name, any attributes and its entire content. The QName dialect  
370 does not support computed values.  
371

### 372 **3.2.2 XPath Level 1 Dialect**

373 The XPath Level 1 expression dialect uses an XPath to reference specific fragments of  
374 the resource representation. The XPath is logically applied to the XML representation  
375 of the resource and the resulting node-set is the resource fragment which is the  
376 subject of the message containing the expression. Table 2 shows an example usage  
377 of this dialect. This dialect is useful for resources with limited XPath processing  
378 capability which do not need to support returning values computed from their  
379 resource representation.

380 The XPath Level 1 dialect is defined in Appendix I – of this specification. An  
381 implementation that uses the XPath Level 1 dialect MUST support the expressions  
382 whose syntax is described by the BNF in Appendix I –. It MAY support additional  
383 expressions defined by XPath 1.0.

384 An XPath Level 1 expression is an expression whose context is:

- 385 • Context Node: the root element of the XML representation of the resource
- 386 • Context Position: 1
- 387 • Context Size: 1
- 388 • Variable Binding: None
- 389 • Node Tests: NameTest and the text NodeType
- 390 • Function Libraries: None
- 391 • Namespace Declarations: Any namespace declarations in-scope where the  
392 XPath expression appears

393 Consider the resource described in Table 1. The XPath Level 1 dialect can define  
394 references to any element, attribute or value in the resource representation.

395 The XPath Level 1 dialect MUST be indicated by using the URI:

396 `http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/Dialect/XPath-`  
397 `Level-1`

398 Expressions in this dialect MUST NOT evaluate to more than a single node. The XPath  
399 Level 1 dialect does not support computed values. Text and attribute nodes MUST  
400 be serialized using the same serialization as for the XPath 1.0 dialect.  
401

### 402 **3.2.3 XPath 1.0 Dialect**

403 The XPath 1.0 expression dialect uses an XPath to reference specific fragments of the  
404 resource representation. The XPath is logically applied to the XML representation of  
405 the resource and the result of the XPath is returned as the value for that expression.  
406 The XPath 1.0 dialect supports a wider set of XPath function libraries than the XPath  
407 Level 1 dialect. Table 7, below, shows an example usage of this dialect. This dialect  
408 is useful for resources with full XPath processing capability or which need to support  
409 returning values computed from their resource representation.

410 An XPath 1.0 expression is an expression whose context is:

- 411 • Context Node: the root element of the XML representation of the resource

- 412 • Context Position: 1
- 413 • Context Size: 1
- 414 • Variable Binding: None
- 415 • Function Libraries: Core function library
- 416 • Namespace Declarations: Any namespace declarations in-scope where the
- 417 XPath expression appears

418 Consider the resource described in Table 1. The XPath 1.0 dialect can define  
419 references to any element, attribute or value in the resource representation and can  
420 also be used to compute values from the resource representation.

421 The XPath 1.0 dialect MUST be indicated by using the URI:

422 <http://www.w3.org/TR/1999/REC-xpath-19991116>

423 Implementations that support the full XPath 1.0 dialect MUST support the XPath  
424 Level 1 dialect.

425 Note that the expression may evaluate to one of four possible types: a node-set, a  
426 Boolean, a number or a string. The latter three types are the results of evaluating a  
427 computed expression. They are serialized by performing the following conversion  
428 and then wrapping the result in the `wsrt:Result` element:

- 429 • Boolean – converted to an `xs:boolean`
- 430 • string – convert to an `xs:string`
- 431 • number – convert to an `xs:double`

432

433 A node-set is zero or more elements, attributes or text values of elements. A node-  
434 set is serialized into XML by concatenating each node and enclosing it in the  
435 `wsrt:Result` wrapper XML element for which schema validation is suppressed.  
436 Element nodes in a node-set are serialized directly into their XML representation.  
437 For attributes and text nodes in the node-set, a wrapper element is used to enclose  
438 these values to distinguish them from other such nodes in the serialized result.

439

440 Attribute nodes in XPath are represented in the following form:

441 `name="value"`

442 Serialization of an attribute node separates the name from the value using the  
443 following element:

444 (01) `<wsrt:AttributeNode name="attribute name">`

445 (02) `attribute value`

446 (03) `</wsrt:AttributeNode>`

447 The following describes additional constraints on the outline listed above:

448 `wsrt:AttributeNode`

449 This element is used to serialize an attribute node in a node-set and MUST  
450 contain the value portion of the attribute node.

451 `wsrt:AttributeNode/@name`

452 This attribute MUST be the name portion of the attribute node.

453

454 Text nodes are serialized in the following form:

455 (01) `<wsrt:TextNode>`

456 (02) `text value`

457 (03) `</wsrt:TextNode>`

458 The following describes additional constraints on the outline listed above:

459 wsrt:TextNode

460 This element is used to serialize a text node in a node-set and MUST contain the  
461 text value.

462

463 Given the following XML as an example document.

```
464 (01) <a xmlns="example">
```

```
465 (02) <b>1</b>
```

```
466 (03) <c x="y">2</c>
```

```
467 (04) </a>
```

468

469 The result of the XPath `"/a/b | /a/b/text() | /a/c/@x"` would be serialized as the  
470 following:

```
471 (01) <wsrt:Result>
```

```
472 (02) <b>1</b>
```

```
473 (03) <wsrt:TextNode>1</wsrt:TextNode>
```

```
474 (04) <wsrt:AttributeNode name="x">y</wsrt:AttributeNode>
```

```
475 (05) </wsrt:Result>
```

476

477 The nodes in the node-set MAY be serialized in any order.

478

479 The WS-RT global element definition `wsrt:NodeSet` can also be used as the wrapper  
480 element when serializing these node-sets outside of a WS-RT result.

481

482 An XPath 1.0 expression may evaluate to multiple nodes; because of this the XPath  
483 1.0 dialect MUST NOT be used with a "Put" or "Create" operation.

### 484 3.3 Get

485 The WS-Transfer Get operation is used to retrieve an existing resource  
486 representation in its entirety. WS-ResourceTransfer extends the "Get" operation to  
487 retrieve fragments of an existing representation. A resource that can return its full  
488 representation MUST also support `wxf:Get` to return the entire resource  
489 representation without using WS-ResourceTransfer extensions.

490 The [Body] of `wsrt:Get` contains an expression that identifies the fragment of interest.

491 The outline for `wsrt:Get` is:

```
492 (01) [Headers]
```

```
493 (02) <wsrt:ResourceTransfer s:mustUnderstand="true"? />
```

```
494 (03) [Action]
```

```
495 (04) http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
```

```
496 (05) [Body]
```

```
497 (06) <wsrt:Get Dialect="xs:anyURI"?>
```

```
498 (07) <wsrt:Expression ...>xs:any</wsrt:Expression> *
```

```
499 (08) </wsrt:Get>
```

500 The following describes additional constraints on the outline listed above:

501 **[Header]**/wsrt:ResourceTransfer

502 If present and understood, a resource MUST process the [Body] in its entirety  
503 and comply with its content. If not present then a resource MUST treat this  
504 request as described in WS-Transfer Get [WS-Transfer].

505 **[Body]/wsrt:Get**

506 An element which controls the retrieval of the resource fragment. This element  
507 MUST be present if the wsrt:ResourceTransfer header is present.

508 **[Body]/wsrt:Get/@Dialect**

509 This URI indicates which dialect expression will be used to identify and retrieve  
510 the fragment(s). This attribute MUST be present when the message contains a  
511 wsrt:Expression element. A resource MUST generate an UnsupportedDialectFault  
512 if it does not support the specified Dialect.

513 **[Body]/wsrt:Get/wsrt:Expression**

514 When present this optional element identifies a fragment in the resource to be  
515 sent in the response. Absence of this element is equivalent to an Expression that  
516 identifies the entire resource representation. The value of this element MUST  
517 conform to the dialect specified in [Body]/wsrt:Get/@Dialect attribute. A  
518 resource MUST generate an InvalidExpressionFault if the expression is invalid. If  
519 the expression syntax is not valid with respect to the dialect then a resource  
520 SHOULD specify a fault detail of "InvalidExpressionSyntax". If the expression  
521 value is not valid for the resource type then the resource SHOULD specify a fault  
522 detail of "InvalidExpressionValue".

523 If a resource cannot return a value for a requested fragment then it MUST  
524 generate a GetFault.

525 If the request contains more Expression elements than the resource supports the  
526 resource MUST return a fault which SHOULD be wsrt:MultipartLimitExceededFault.

527 If the resource accepts a wsrt:Get request and processes it successfully it MUST  
528 reply with a response of the following form:

529 **(01) [Headers]**

530 **(02)** <wsrt:ResourceTransfer/>

531 **(03) [Action]**

532 **(04)** http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse

533 **(05) [Body]**

534 **(06)** <wsrt:GetResponse>

535 **(07)** <wsrt:Result...>xs:any</wsrt:Result> +

536 **(08)** </wsrt:GetResponse>

537 The following describes additional constraints on the outline listed above:

538 **[Headers]/wsrt:ResourceTransfer**

539 This header indicates that the response contains body content defined in WS-  
540 ResourceTransfer.

541 **[Body]/wsrt:GetResponse**

542 An element which wraps the packaging of the fragments in the response. This  
543 element MUST be present if the request Body contained a wsrt:Get element.

544 **[Body]/wsrt:GetResponse/wsrt:Result**

545 This element encompasses a single fragment response corresponding to a  
546 wsrt:Expression in the original request and MUST contain the fragment of the  
547 resource representation identified by the wsrt:Expression. If the request  
548 contained no wsrt:Expression then this element MUST contain the entire resource  
549 representation. If the request contained one or more wsrt:Expression elements  
550 then for each wsrt:Expression element in the request there MUST be one

551 wsrt:Result element in the response even if the wsrt:Result has empty content.  
552 The wsrt:Result elements MUST appear in the same order in the response as the  
553 corresponding wsrt:Expression elements in the request. A wsrt:Result MUST have  
554 empty content in the case where a wsrt:Expression resolves to nothing.  
555

556 An example Get message using the XPath Level 1 dialect is shown above in Table 2  
557 and the expected GetResponse is shown in Table 3, for the resource whose XML  
558 representation is shown in Table 1 above.

559 An example Get message that uses the QName dialect is shown in Table 5 below.  
560

561 **Table 5: Example Get message using the "QName" dialect**

```
562 (01) <s:Envelope
563 (02)     xmlns:s="http://www.w3.org/2003/05/soap-envelope"
564 (03)     xmlns:wsa="http://www.w3.org/2005/08/addressing"
565 (04)     xmlns:wsrt=
566 (05)         "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer">
567 (06) <s:Header>
568 (07)     <wsa:To>http://www.example.org/disk</wsa:To>
569 (08)     <wsa:Action s:mustUnderstand="true">
570 (09)         http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
571 (10)     </wsa:Action>
572 (11)     <wsrt:ResourceTransfer s:mustUnderstand="true"/>
573 (12)     ...
574 (13) </s:Header>
575 (14) <s:Body>
576 (15)     <wsrt:Get Dialect="http://schemas.xmlsoap.org/ws/2006/08/
577 (16)         resourceTransfer/Dialect/QName"
578 (17)     xmlns:d="http://example.org/sample">
579 (18)         <wsrt:Expression>
580 (19)             d:Volume
581 (20)         </wsrt:Expression>
582 (21)         <wsrt:Expression>
583 (22)             d:DiskCapacity
584 (23)         </wsrt:Expression>
585 (24)     </wsrt:Get>
586 (25) </s:Body>
587 (26) </s:Envelope>
```

588 The fragments of the resource representation are identified by the wsrt:Expression  
589 contents on lines (19) and (22). Notice that the *d:Volume* QName in the example  
590 resolves to three elements in the resource representation. The response to this "Get"  
591 message is illustrated in Table 6.

592 **Table 6: Example GetResponse using the "QName" dialect**

```
593 (01) <s:Envelope
594 (02)     xmlns:s="http://www.w3.org/2003/05/soap-envelope"
595 (03)     xmlns:wsa="http://www.w3.org/2005/08/addressing"
596 (04)     xmlns:wsrt=
```

```

597 (05) "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer">
598 (06) <s:Header>
599 (07) <wsa:To>
600 (08) http://www.w3.org/2005/08/addressing/anonymous
601 (09) </wsa:To>
602 (10) <wsa:Action s:mustUnderstand="true">
603 (11) http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
604 (12) </wsa:Action>
605 (13) <wsrt:ResourceTransfer/>
606 (14) ...
607 (15) </s:Header>
608 (16) <s:Body>
609 (17) <wsrt:GetResponse xmlns:d="http://example.org/sample">
610 (18) <wsrt:Result>
611 (19) <d:Volume>
612 (20) <d:Drive>C:</d:Drive>
613 (21) <d:Label>MyDrive-C</d:Label>
614 (22) <d:TotalCapacity>1000000000</d:TotalCapacity>
615 (23) <d:FreeSpace>6234794528</d:FreeSpace>
616 (24) </d:Volume>
617 (25) <d:Volume>
618 (26) <d:Drive>D:</d:Drive>
619 (27) <d:Label>MyDrive-D</d:Label>
620 (28) <d:TotalCapacity>3000000000</d:TotalCapacity>
621 (29) <d:FreeSpace>26462809800</d:FreeSpace>
622 (30) </d:Volume>
623 (31) <d:Volume>
624 (32) <d:Drive>E:</d:Drive>
625 (33) <d:Label>MyDrive-E</d:Label>
626 (34) <d:TotalCapacity>2250000000</d:TotalCapacity>
627 (35) <d:FreeSpace>16056784170</d:FreeSpace>
628 (36) </d:Volume>
629 (37) </wsrt:Result>
630 (38) <wsrt:Result>
631 (39) <d:DiskCapacity>6250000000</d:DiskCapacity>
632 (40) </wsrt:Result>
633 (41) </wsrt:GetResponse>
634 (42) </s:Body>
635 (43) </s:Envelope>

```

636 The value of each of the wsrt:Expression fragments in the request message is  
637 returned in a unique wsrt:Result wrapper in the response message. The order of the  
638 wsrt:Result elements in the response matches the order of the corresponding  
639 wsrt:Expression elements in the request. The result of getting the <d:Volume>  
640 fragment, for example, is shown on lines (18) - (37).

641

642 One final example of the Get operation, using the full XPath 1.0 dialect, is shown  
643 below in Table 7. This illustrates the use of a query expression to return the  
644 computed result of the quantity of *d:Volume* elements that have a capacity greater  
645 than 20000000000. For the sake of brevity, only the message body is shown.

646 **Table 7: Example Get message using an XPath 1.0 query expression**

```
647 (01) <s:Body>
648 (02)   <wsrt:Get Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116"
649 (03)     xmlns:d="http://example.org/sample">
650 (04)       <wsrt:Expression>
651 (05)         count( d:Volume[d:TotalCapacity > 20000000000] )
652 (06)       </wsrt:Expression>
653 (07)   </wsrt:Get>
654 (08) </s:Body>
```

655 The expression on line (05), when applied to the resource representation shown in  
656 Table 1, evaluates to a result of "2".

657 **Table 8: Example GetResponse of an XPath 1.0 query expression**

```
658 (01) <s:Body>
659 (02)   <wsrt:GetResponse>
660 (03)     <wsrt:Result>2</wsrt:Result>
661 (04)   </wsrt:GetResponse>
662 (05) </s:Body>
```

663 The result of the query expression is a number which is converted to an *xs:double*  
664 and returned as the value of the *wsrt:Result* element shown on line (03).

665

### 666 3.4 Put

667 The WS-Transfer "Put" operation is used to update an existing resource  
668 representation by providing a replacement XML representation. WS-ResourceTransfer  
669 extends the "Put" operation to update an existing resource representation by  
670 providing fragments of the XML representation. A resource that can update its full  
671 representation MUST also support *wxf:Put* to update the entire resource  
672 representation without using WS-ResourceTransfer extensions.

673 The extended outline for the "Put" operation is:

```
674 (01) [Headers]
675 (02) <wsrt:ResourceTransfer s:mustUnderstand="true"/>
676 (03) [Action]
677 (04) http://schemas.xmlsoap.org/ws/2004/09/transfer/Put
678 (05) [Body]
679 (06) <wsrt:Put Dialect="xs:anyURI"?>
680 (07)   <wsrt:Fragment Mode="Modify|Insert|Remove">
681 (08)     <wsrt:Expression>xs:any</wsrt:Expression> ?
682 (09)     <wsrt:Value ...>xs:any</wsrt:Value> ?
683 (10)   </wsrt:Fragment> +
684 (11) </wsrt:Put>
```

685 The following describes additional constraints on the outline listed above:

686 **[Header]**/wsrt:ResourceTransfer



687 If present and understood, a resource MUST process the [Body] in its entirety  
688 and comply with its content. If not present then a resource MUST treat this  
689 request as described in WS-Transfer Put [WS-Transfer].

690 **[Body]/wsrt:Put**  
691 An element that specifies the fragments of the resource representation to update.  
692 This element MUST be present if the wsrt:ResourceTransfer header is present.

693 **[Body]/wsrt:Put/@Dialect**  
694 This URI indicates which expression dialect will be used to identify the  
695 fragment(s) of the resource representation to be updated. This attribute MUST be  
696 present when the message contains a wsrt:Expression element.

697 **[Body]/wsrt:Put/Fragment**  
698 This element encompasses a single update to be performed on the resource.  
699 Upon successful completion of a Put operation, the resource representation MUST  
700 appear as though the fragment updates occurred in the order specified in the Put  
701 operation. If there are multiple Fragment elements then, for the first fragment,  
702 the resource representation is the original resource representation (before  
703 applying the Put changes). For subsequent fragments, the resource  
704 representation is the intermediate representation resulting from applying the  
705 previous fragments.

706 If the request contains more Fragment elements than the resource supports the  
707 resource MUST return a fault which SHOULD be wsrt:MultipartLimitExceededFault.

708 **[Body]/wsrt:Put/Fragment/@Mode**  
709 This attribute indicates the type of update to be performed on this fragment. A  
710 resource MUST generate a ResourceValidityFault if the result of executing this  
711 operation would cause the resource representation to become invalid. A resource  
712 MAY support only a subset of these Modes. A resource that does not support a  
713 specified Mode MUST generate a PutModeUnsupportedFault.

714 A value of "Remove" indicates that the fragment MUST be deleted if it is present.  
715 The expression dialect indicated in this operation MAY place additional constraints  
716 on the definition of this Mode. Note that, in order to delete the resource itself, a  
717 WS-Transfer "Delete" message is used.

718 A value of "Modify" means that the fragment MUST be replaced by removing any  
719 fragment that already exists and inserting the specified value in its place. If the  
720 expression resolves to nothing then this fragment element does not result in any  
721 change to the resource representation. The expression dialect indicated in this  
722 operation MAY place additional constraints on the definition of this Mode. A  
723 fragment with no wsrt:Expression MUST specify this Mode.

724 A value of "Insert" indicates that the fragment MUST be added to the resource  
725 representation. If the expression targets a repeated element (maxOccurs > 1),  
726 the fragment MUST be added at the end. If the expression targets a non-  
727 repeated element (maxOccurs = 1) that already exists, the resource MUST  
728 generate a FragmentAlreadyExistsFault. If the expression targets an existing  
729 item of a repeated element, the fragment MUST be added before the existing  
730 item.

731 **[Body]/wsrt:Put/Fragment/Expression**  
732 When present this optional element contains the expression that identifies a  
733 fragment of the resource representation to be updated. Absence of this element  
734 is equivalent to an Expression that identifies the entire resource representation.

735 The value of this element MUST conform to the dialect specified in the  
736 [Body]/wsrt:Put/@Dialect attribute. A resource MUST generate an

737 InvalidExpressionFault if the expression is invalid. If the expression syntax is not  
738 valid with respect to the dialect then a resource SHOULD specify a fault detail of  
739 "InvalidExpressionSyntax". If the expression value is not valid for the resource  
740 type then the resource SHOULD specify a fault detail of "InvalidExpressionValue".

741 **[Body]**/wsrt:Put/Fragment/Value

742 This element contains the data to be written to the resource representation. If  
743 the [Body]/wsrt:Put/Fragment/@Mode attribute is "Insert" or "Modify" then this  
744 element MUST be present. If the [Body]/wsrt:Put/Fragment/@Mode attribute is  
745 "Remove" then this element MUST NOT be present. A resource MUST generate an  
746 InvalidPutSyntaxFault if it receives a message with a Value cardinality that is not  
747 valid for the Mode attribute.  
748

749 If a resource encounters a failure while processing the fragments in a Put request, it  
750 MUST generate a PutFault. The resource SHOULD ensure that its representation is  
751 unchanged from prior to the request, although atomic behavior is not required of  
752 resource implementations. The resource SHOULD include a wsrt:SideAffects element  
753 in the fault detail to indicate whether any changes occurred.

754 If the resource accepts a Put request and performs the requested update, it MUST  
755 reply with a response of the following form:

756 (01) **[Headers]**

757 (02) <wsrt:ResourceTransfer/>

758 (03) **[Action]**

759 (04) http://schemas.xmlsoap.org/ws/2004/09/transfer/PutResponse

760 (05) **[Body]**

761

762 The following describes additional constraints on the outline listed above:

763 **[Headers]**/wsrt:ResourceTransfer

764 This header indicates that the response contains body content defined in WS-  
765 ResourceTransfer.

766 **[Body]**

767 If the request Body contained a wsrt:Put element then the new representation  
768 MUST be omitted in the response. Otherwise the response MUST be as described  
769 in WS-Transfer. The absence of the resource representation in the response is in  
770 recognition of the potentially large amount of data that may be returned, which  
771 may have been the reason a fragment Put was used instead of sending the entire  
772 resource representation.  
773

774 An example Put message using the XPath Level 1 dialect is shown in Table 9. For  
775 brevity only the message body is shown.

776 **Table 9 – Example Put message using the XPath Level 1 dialect**

777 (01) <s:Body>

778 (02) <wsrt:Put Dialect="http://schemas.xmlsoap.org/ws/2006/08/  
779 (03) resourceTransfer/Dialect/XPath-Level-1"

780 (04) xmlns:d="http://example.org/sample">

781 (05) <wsrt:Fragment Mode="Remove">

782 (06) <wsrt:Expression>

783 (07) d:Volume[1]

784 (08) </wsrt:Expression>

```

785 (09) </wsrt:Fragment>
786 (10) <wsrt:Fragment Mode="Insert">
787 (11) <wsrt:Expression>
788 (12)   d:Volume[2]
789 (13) </wsrt:Expression>
790 (14) <wsrt:Value>
791 (15)   <d:Volume>
792 (16)     <d:Drive>X:</d:Drive>
793 (17)     <d:Label>MyDrive-X</d:Label>
794 (18)     <d:TotalCapacity>5000000000</d:TotalCapacity>
795 (19)   </d:Volume>
796 (20) </wsrt:Value>
797 (21) </wsrt:Fragment>
798 (22) </wsrt:Put>
799 (23) </s:Body>

```

800 Line (05) indicates that a fragment should be removed and is targeted at the 1<sup>st</sup>  
801 Volume element, identified by the wsrt:Expression contents on line (06). Line (10)  
802 indicates that a fragment should be inserted into the representation that results from  
803 applying the first fragment update. The insertion location is identified by the  
804 wsrt:Expression contents on line (12), i.e immediately before the second Volume  
805 element. Lines (14) - (19) show the content of the new Volume which is to be  
806 inserted. The updated resource representation is illustrated in Table 10.

807 **Table 10: Updated resource representation**

```

808 (01) <Disk xmlns="http://example.org/sample">
809 (02)   <DiskCapacity>6250000000</DiskCapacity>
810 (03)   <DiskFreeSpace>524182841</DiskFreeSpace>
811 (04)   <SerialNumber>123-F2560</SerialNumber>
812 (05)   <LastAuditDate>1998-05-25T13:30:15</LastAuditDate>
813 (06)   <Volume>
814 (07)     <Drive>D:</Drive>
815 (08)     <Label>MyDrive-D</Label>
816 (09)     <TotalCapacity>3000000000</TotalCapacity>
817 (10)     <FreeSpace>26462809800</FreeSpace>
818 (11)   </Volume>
819 (12)   <Volume>
820 (13)     <Drive>X:</Drive>
821 (14)     <Label>MyDrive-X</Label>
822 (15)     <TotalCapacity>5000000000</TotalCapacity>
823 (16)     <FreeSpace>5000000000</FreeSpace>
824 (17)   </Volume>
825 (18)   <Volume>
826 (19)     <Drive>E:</Drive>
827 (20)     <Label>MyDrive-E</Label>
828 (21)     <TotalCapacity>22500000000</TotalCapacity>
829 (22)     <FreeSpace>16056784170</FreeSpace>
830 (23)   </Volume>

```

831 (24) </Disk>

832 Lines (12) - (17) show the result of the Put@Insert operation. Drive C was removed  
833 and X was inserted at position 2 in between drive D and E since the expression  
834 targeted the 2<sup>nd</sup> Volume element after the removal of C.

835

836 An example Put message using the QName dialect is shown in Table 11. For brevity  
837 only the message body is shown.

838 **Table 11 – Example Put message using the QName dialect**

839 (01) <s:Body>

840 (02) <wsrt:Put Dialect="http://schemas.xmlsoap.org/ws/2006/08/

841 (03) resourceTransfer/Dialect/QName "

842 (04) xmlns:d="http://example.org/sample">

843 (05) <wsrt:Fragment Mode="Modify">

844 (06) <wsrt:Expression>

845 (07) d:Volume

846 (08) </wsrt:Expression>

847 (09) <wsrt:Value>

848 (10) <d:Volume>

849 (11) <d:Drive>F:</d:Drive>

850 (12) <d:Label>MyDrive-F</d:Label>

851 (13) <d:TotalCapacity>5000000000</d:TotalCapacity>

852 (14) </d:Volume>

853 (15) <d:Volume>

854 (16) <d:Drive>D:</d:Drive>

855 (17) <d:Label>MyDrive-D</d:Label>

856 (18) <d:TotalCapacity>3000000000</d:TotalCapacity>

857 (19) </d:Volume>

858 (20) </wsrt:Value>

859 (21) </wsrt:Fragment>

860 (22) <wsrt:Fragment Mode="Insert">

861 (23) <wsrt:Expression>

862 (24) d:Volume

863 (25) </wsrt:Expression>

864 (26) <wsrt:Value>

865 (27) <d:Volume>

866 (28) <d:Drive>X:</d:Drive>

867 (29) <d:Label>MyDrive-X</d:Label>

868 (30) <d:TotalCapacity>5000000000</d:TotalCapacity>

869 (31) </d:Volume>

870 (32) </wsrt:Value>

871 (33) </wsrt:Fragment>

872 (34) </wsrt:Put>

873 (35) </s:Body>

874 Line (05) again indicates that the fragment needs to be updated (i.e logically  
875 removed and then replaced). The target fragment of the resource is the set of

876 d: Volume elements, identified by the wsrt:Expression contents on line (07). Lines  
877 (09) - (20) show the new value for this set of elements to use to replace the old set.  
878 Lines (22) - (33) indicates that a fragment should be inserted into the representation  
879 that results from applying the first fragment update. Lines (26) - (32) show the  
880 content of the new Volume which is added at the end.

881 The updated resource representation is illustrated in Table 12.

882 **Table 12: Updated resource representation**

```
883 (01) <Disk xmlns="http://example.org/sample">  
884 (02)   <DiskCapacity>6250000000</DiskCapacity>  
885 (03)   <DiskFreeSpace>524182841</DiskFreeSpace>  
886 (04)   <SerialNumber>123-F2560</SerialNumber>  
887 (05)   <LastAuditDate>1998-05-25T13:30:15</LastAuditDate>  
888 (06)   <Volume>  
889 (07)     <Drive>F:</Drive>  
890 (08)     <Label>MyDrive-F</Label>  
891 (09)     <TotalCapacity>5000000000</TotalCapacity>  
892 (10)     <FreeSpace>5000000000</FreeSpace>  
893 (11)   </Volume>  
894 (12)   <Volume>  
895 (13)     <Drive>D:</Drive>  
896 (14)     <Label>MyDrive-D</Label>  
897 (15)     <TotalCapacity>3000000000</TotalCapacity>  
898 (16)     <FreeSpace>3000000000</FreeSpace>  
899 (17)   </Volume>  
900 (18)   <Volume>  
901 (19)     <Drive>X:</Drive>  
902 (20)     <Label>MyDrive-X</Label>  
903 (21)     <TotalCapacity>5000000000</TotalCapacity>  
904 (22)     <FreeSpace>5000000000</FreeSpace>  
905 (23)   </Volume>  
906 (24) </Disk>
```

907 Lines (06) - (17) show the result of the Put@Modify operation. Drives C, D and E are  
908 replaced by drives D and F. Effectively, drives C and E are removed and drive F is  
909 inserted. Lines (18) - (23) show the result of the Put@Insert operation. The Volume  
910 element for drive X is added at the end of the array of Volumes.

911

### 912 **3.5 Create**

913 The WS-Transfer "Create" operation is used for creating a resource via an initial  
914 representation. The resource factory that receives a Create request will allocate a  
915 new resource that is initialized from the presented representation. The new resource  
916 will be assigned a factory-service-determined endpoint reference that is returned in  
917 the response message. In many cases, the information required to create a resource  
918 may markedly differ from the initial representation (the value as realized by a  
919 subsequent "Get" operation), and supplying the initial representation is not viable.

920 WS-ResourceTransfer extends the "Create" operation to create a resource from zero  
921 or more specified fragments of the XML representation. WS-ResourceTransfer further

922 extends the "Create" operation such that any resource metadata MAY be created as  
923 part of the creation of the resource.

924

925 The extended outline for the "Create" operation is:

```
926 (01) [Headers]
927 (02) <wsrt:ResourceTransfer s:mustUnderstand="true"/>
928 (03) [Action]
929 (04) http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
930 (05) [Body]
931 (06) <wsrt:Create Dialect="xs:anyURI"?>
932 (07) <wsmex:Metadata>resource metadata</wsmex:Metadata> ?
933 (08) <wsrt:Fragment>
934 (09) <wsrt:Expression>xs:any</wsrt:Expression> ?
935 (10) <wsrt:Value ...>xs:any</wsrt:Value>
936 (11) </wsrt:Fragment> *
937 (12) </wsrt:Create>
```

938 The following describes additional constraints on the outline listed above:

939 **[Header]**/wsrt:ResourceTransfer

940 If present and understood, a resource MUST process the [Body] in its entirety  
941 and comply with its content. If not present then a resource MUST treat this  
942 request as described in WS-Transfer Create [WS-Transfer].

943 **[Body]**/wsrt:Create

944 An element that specifies the fragments of the resource representation to be  
945 initialized during resource creation and optionally any resource metadata that is  
946 to be created as part of the creation of the resource. This element MUST be  
947 present if the wsrt:ResourceTransfer header is present.

948 **[Body]**/wsrt:Create/@Dialect

949 This URI indicates which expression dialect will be used to identify the  
950 fragment(s) of the resource representation to be initialized during resource  
951 creation. This attribute MUST be present when the message contains a  
952 wsrt:Expression element.

953 **[Body]**/wsrt:Create/wsmex:Metadata

954 When present this optional element MUST contain at least one  
955 wsmex:MetadataSection. This is resource metadata to be created and initialized  
956 during the creation of the resource.

957 A resource factory MUST generate an InvalidMetadataFault if the Create request  
958 message contains a wsmex:Dialect that is not supported or if the resource  
959 metadata contains values that are not supported for the resource.

960 This element MAY contain a wsmex:MetadataSection with a wsmex:Dialect of  
961 http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer allowing the  
962 requestor to specify desired metadata as defined in this specification (such as  
963 lifecycle metadata).

964 **[Body]**/wsrt:Create/Fragment

965 This element encompasses a single resource fragment to be initialized during the  
966 resource creation. If there are multiple Fragment elements then the resource  
967 MUST appear to have been created as though each fragment were processed in  
968 the sequence specified in the Create message.

969 If the request contains more Fragment elements than the resource supports the  
970 resource MUST return a fault which SHOULD be wsrt:MultipartLimitExceededFault.

971 **[Body]/wsrt:Create/Fragment/Expression**

972 When present this optional element contains an expression that identifies a  
973 resource fragment to be initialized during resource creation. The expression  
974 identifies the fragment in the resource representation as it appears *after*  
975 successful processing of the current fragment. Absence of this element is  
976 equivalent to an Expression that identifies the entire resource representation. The  
977 value of this element MUST conform to the dialect specified in the  
978 **[Body]/wsrt:Create/@Dialect** attribute. A resource factory MUST generate an  
979 **InvalidExpressionFault** if the expression is invalid. If the expression syntax is not  
980 valid with respect to the dialect then a resource factory SHOULD specify a fault  
981 detail of "InvalidExpressionSyntax". If the expression is not valid for the resource  
982 type then the resource factory SHOULD specify a fault detail of  
983 "InvalidExpressionValue".

984 **[Body]/wsrt:Create/Fragment/Value**

985 This element contains the data to be written to the resource representation. If  
986 the resource factory is unable to write the requested fragment then it MUST  
987 generate a **CreateFault**.

988 If the resource factory accepts a Create request, it MUST reply with a response of  
989 the following form:

990 (01) **[Headers]**

991 (02) <wsrt:ResourceTransfer/>

992 (03) **[Action]**

993 (04) http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse

994 (05) **[Body]**

995 (06) <wxf:ResourceCreated>

996 (07) wsa:EndpointReferenceType

997 (08) </wxf:ResourceCreated>

998 The following describes additional constraints on the outline listed above:

999 **[Headers]/wsrt:ResourceTransfer**

1000 This header indicates that the response contains body content defined in WS-  
1001 ResourceTransfer.

1002 **[Body]/wxf:ResourceCreated**

1003 This element contains the endpoint reference for the resource that was created.  
1004 All subsequent access to the resource MUST be done using this EPR.

1005 If the request Body contained a wsrt:Create element then the new representation  
1006 MUST be omitted in the response. Otherwise the response MUST be as described in  
1007 WS-Transfer.

1008

1009 An example Create message using the QName dialect is shown in Table 13. For  
1010 brevity only the message body is shown.

1011 **Table 13 - Example Create message using the QName dialect**

1012 (01) <s:Body>

1013 (02) <wsrt:Create Dialect="http://schemas.xmlsoap.org/ws/2006/08/

1014 (03) resourceTransfer/Dialect/QName"

1015 (04) xmlns:d="http://example.org/sample">

1016 (05) <ws mex:Metadata>

```

1017 (06) <wsmex:MetadataSection Dialect="http://schemas.xmlsoap.org/
1018 (07) ws/2006/08/resourceTransfer">
1019 (08) <wsrt:Metadata>
1020 (09) <wsrt:Lifetime>
1021 (10) <wsrt:TerminateAt>
1022 (11) <wsrt:TerminationTime>
1023 (12) 2006-04-11T12:00:00Z
1024 (13) </wsrt:TerminationTime>
1025 (14) <wsrt:CurrentTime>
1026 (15) 2006-04-10T10:00:54Z
1027 (16) </wsrt:CurrentTime>
1028 (17) </wsrt:TerminateAt>
1029 (18) </wsrt:Lifetime>
1030 (19) </wsrt:Metadata>
1031 (20) </wsmex:MetadataSection>
1032 (21) </wsmex:Metadata>
1033 (22) <wsrt:Fragment>
1034 (23) <wsrt:Expression>
1035 (24) d:Volume
1036 (25) </wsrt:Expression>
1037 (26) <wsrt:Value>
1038 (27) <d:Volume>
1039 (28) <d:Drive>C:</d:Drive>
1040 (29) <d:Label>MyDrive-C</d:Label>
1041 (30) <d:TotalCapacity>10000000000</d:TotalCapacity>
1042 (31) </d:Volume>
1043 (32) <d:Volume>
1044 (33) <d:Drive>D:</d:Drive>
1045 (34) <d:Label>MyDrive-D</d:Label>
1046 (35) <d:TotalCapacity>30000000000</d:TotalCapacity>
1047 (36) </d:Volume>
1048 (37) </wsrt:Value>
1049 (38) </wsrt:Fragment>
1050 (39) </wsrt:Create>
1051 (40) </s:Body>

```

1052

1053 Line (10) indicates that the resource, once created, is scheduled for destruction at  
1054 the specific time. Messages sent to the EPR returned in the CreateResponse, after  
1055 this time, will fault. Line (24) indicates that resource is created with a specific value  
1056 or set of values for the <d:Volume> property. Lines (26) - (37) specify the set of  
1057 values of the <d:Volume> property. The response to this "Create" message is  
1058 illustrated in Table 14.

1059 **Table 14: Example CreateResponse**

```

1060 (01) <s:Body>
1061 (02) <wxf:ResourceCreated>
1062 (03) <wsa:Address>http://www.example.org/diskport</wsa:Address>

```



```

1063 (04) <wsa:ReferenceParameters>
1064 (05) <xyz:ManagedResource>44355</xyz:ManagedResource>
1065 (06) </wsa:ReferenceParameters>
1066 (07) </wxf:ResourceCreated>
1067 (08) </s:Body>

```

1068  
1069 Lines (02) - (07) show the EPR to the disk resource that is returned in the response  
1070 message.  
1071

## 1072 4. Faults

1073 WS-ResourceTransfer faults MUST include as the [Action] property the following fault  
1074 action URI:

```
1075 http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
```

1076 The faults defined in this section are generated if the condition stated in the  
1077 preamble is met. Faults are targeted at a destination endpoint according to the fault  
1078 handling rules defined in [\[WS-Addressing\]](#).

1079 The definitions of faults in this section use the following properties:

1080 [Code] The fault code.

1081 [Subcode] The fault subcode.

1082 [Reason] The English language reason element.

1083 [Detail] The detail element. If absent, no detail element is defined for the fault.

1084 For SOAP 1.2, the [Code] property MUST be either "Sender" or "Receiver". These  
1085 properties are serialized into text XML as follows:

1086

SOAP Version	Sender	Receiver
SOAP 1.2	s12:Sender	s12:Receiver

1087

1088 The properties above bind to a SOAP 1.2 fault as follows:

```

1089 <s12:Envelope>
1090 <s12:Header>
1091 <wsa:Action>
1092 http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
1093 </wsa:Action>
1094 <!-- Headers elided for clarity. -->
1095 </s12:Header>
1096 <s12:Body>
1097 <s12:Fault>
1098 <s12:Code>
1099 <s12:Value>[Code]</s12:Value>
1100 <s12:Subcode>
1101 <s12:Value>[Subcode]</s12:Value>
1102 </s12:Subcode>

```

```
1103 </s12:Code>
1104 <s12:Reason>
1105   <s12:Text xml:lang="en">[Reason]</s12:Text>
1106 </s12:Reason>
1107 <s12:Detail>
1108   [Detail]
1109   ...
1110 </s12:Detail>
1111 </s12:Fault>
1112 </s12:Body>
1113 </s12:Envelope>
```

1114

1115 The properties bind to a SOAP 1.1 fault as follows:

```
1116 <s11:Envelope>
1117 <s11:Body>
1118 <s11:Fault>
1119 <faultcode>[Subcode]</faultcode>
1120 <faultstring xml:lang="en">[Reason]</faultstring>
1121 <detail>
1122   [Detail]
1123   ...
1124 </detail>
1125 </s11:Fault>
1126 </s11:Body>
1127 </s11:Envelope>
```

1128

#### 1129 **4.1 wsa:DestinationUnreachable**

1130 This fault is sent in response to a message that is targeted at a resource that cannot  
1131 be found and is deemed not to exist. This may be because the resource was never  
1132 created or because the resource has been destroyed – there is no distinction  
1133 between these cases.

1134 The SOAP bindings for this fault are defined in [\[WS-Addressing\]](#).

#### 1135 **4.2 wsa:EndpointUnavailable**

1136 The resource is unable to process the message at this time due to some transient  
1137 issue. The endpoint MAY optionally include a wsa:RetryAfter parameter in the detail.  
1138 The source should not retransmit the message until this duration has passed.

1139 The SOAP bindings for this fault are defined in [\[WS-Addressing\]](#).

#### 1140 **4.3 ConcurrencyFault**

1141 This fault is generated by a resource to indicate that it was unable to process a  
1142 message due to concurrent access. A requester might choose to handle this condition  
1143 by retrying the operation that caused it.

1144

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Sender
<b>[Subcode]</b>	wsrt:ConcurrencyFault
<b>[Reason]</b>	Could not access the resource due to concurrency and/or locking conditions
<b>[Detail]</b>	

#### 1145 4.4 UnsupportedDialectFault

1146 This fault is generated by a resource to indicate that the expression dialect used to  
 1147 identify a resource fragment is not supported by the resource for the current  
 1148 operation. The fault detail SHOULD contain the Dialect values that the resource does  
 1149 support for the operation.

1150

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Sender
<b>[Subcode]</b>	wsrt:UnsupportedDialectFault
<b>[Reason]</b>	The requested dialect is not supported
<b>[Detail]</b>	<wsrt:Dialect>xs:anyURI</wsrt:Dialect> *

#### 1151 4.5 InvalidExpressionFault

1152 This fault is sent by a resource if a <wsrt:Expression> element has an syntax that is  
 1153 invalid according to the definition of the expression dialect. If the expression syntax  
 1154 is not valid with respect to the dialect then a resource SHOULD specify a fault detail  
 1155 of "InvalidExpressionSyntax", indicating which expression this detail applies to. If the  
 1156 expression is not valid for the resource type then the resource SHOULD specify a  
 1157 fault detail of "InvalidExpressionValue", indicating which expression this detail  
 1158 applies to.

1159

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Sender
<b>[Subcode]</b>	wsrt:InvalidExpressionFault
<b>[Reason]</b>	The specified Expression is not valid
<b>[Detail]</b>	<pre> &lt;wsrt:InvalidExpressionSyntax&gt;   &lt;wsrt:Expression&gt;xs:any&lt;/wsrt:Expression&gt; + &lt;/wsrt:InvalidExpressionSyntax&gt;   &lt;wsrt:InvalidExpressionValue&gt;   &lt;wsrt:Expression&gt;xs:any&lt;/wsrt:Expression&gt; + &lt;/wsrt:InvalidExpressionValue&gt; </pre>

1160

#### 1161 4.6 GetFault

1162 This fault is generated by a resource if it is unable to process a valid Get message.

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
-----------------	--

<b>[Code]</b>	s12:Receiver
<b>[Subcode]</b>	wsrt:GetFault
<b>[Reason]</b>	Unable to process Get message
<b>[Detail]</b>	

1163

#### 1164 4.7 ResourceValidityFault

1165 This fault is generated by a resource if the result of processing a Put message would  
 1166 cause the resource representation to become invalid. The fault detail MAY include the  
 1167 wsrt:Fragment element in the Put message that caused this fault to be generated.

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Sender
<b>[Subcode]</b>	wsrt:ResourceValidityFault
<b>[Reason]</b>	The requested resource modification is not valid.
<b>[Detail]</b>	<wsrt:Fragment>fragment</wsrt:Fragment> ?

1168

#### 1169 4.8 FragmentAlreadyExistsFault

1170 This fault is generated by a resource if a "Put" message specifies the "Insert" mode  
 1171 and identifies a non-repeated fragment element (maxOccurs = 1) that already  
 1172 exists. The fault detail MAY include the wsrt:Fragment that failed to be processed.

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Sender
<b>[Subcode]</b>	wsrt:FragmentAlreadyExistsFault
<b>[Reason]</b>	The fragment already exists
<b>[Detail]</b>	<wsrt:Fragment>fragment</wsrt:Fragment> ?

#### 1173 4.9 PutFault

1174 This fault is generated by a resource if it is unable to process a valid Put message.  
 1175 The fault detail MAY include the wsrt:Fragment that failed to be processed.  
 1176 The fault detail SHOULD include a wsrt:SideEffects element in the fault detail to  
 1177 indicate whether any changes occurred. A value of "true" indicates some changes  
 1178 occurred; a value of "false" indicates no changes occurred. Absence of the element  
 1179 indicates that changes may have occurred.

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Receiver
<b>[Subcode]</b>	wsrt:PutFault
<b>[Reason]</b>	Unable to process Put message
<b>[Detail]</b>	<wsrt:Fragment>fragment</wsrt:Fragment> ? <wsrt:SideEffects>xs:boolean</wsrt:SideEffects> ?

1180

1181 **4.10 PutModeUnsupportedFault**

1182 This fault is generated by a resource if a "Put" message specifies a mode that is not  
 1183 supported by the resource.

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Sender
<b>[Subcode]</b>	wsrt:PutModeUnsupportedFault
<b>[Reason]</b>	The Put mode is not supported
<b>[Detail]</b>	

1184

1185 **4.11 CreateFault**

1186 This fault is generated by a resource if it is unable to process a valid Create message.  
 1187 The fault detail MAY include the wsrt:Fragment that failed to be processed.

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Receiver
<b>[Subcode]</b>	wsrt:CreateFault
<b>[Reason]</b>	Unable to process Create message
<b>[Detail]</b>	<wsrt:Fragment> <i>fragment</i> </wsrt:Fragment> ?

1188

1189 **4.12 InvalidMetadataFault**

1190 This fault is generated by a resource factory if a "Create" message contains a  
 1191 wsmex:Dialect that is not supported or if the resource metadata contains values  
 1192 that are not supported for the resource.

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Sender
<b>[Subcode]</b>	wsrt:InvalidMetadataFault
<b>[Reason]</b>	Resource metadata values not supported by resource
<b>[Detail]</b>	

1193

1194 **4.13 MultipartLimitExceededFault**

1195 This fault is generated by a resource if a request message exceeds the limit of  
 1196 wsrt:Expression or wsrt:Fragment elements supported for the dialect. The fault  
 1197 detail MUST contain the maximum number of wsrt:Expression or wsrt:Fragment  
 1198 elements supported for the dialect.

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Sender
<b>[Subcode]</b>	wsrt:MultipartLimitExceededFault
<b>[Reason]</b>	Access to multiple fragments exceeded the supported number of fragments in a single message
<b>[Detail]</b>	<wsrt:MultipartLimit> <i>xs:positiveInteger</i> </wsrt:MultipartLimit>

1199

#### 1200 **4.14 InvalidPutSyntaxFault**

1201 This fault is generated by a resource if a Put request specifying a Mode "Remove"  
1202 contains a wsrt:Value element or if a Put request specifying a Mode "Insert" or  
1203 "Modify" does not contain a wsrt:Value element.

1204

<b>[Action]</b>	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
<b>[Code]</b>	s12:Sender
<b>[Subcode]</b>	wsrt:InvalidRemoveSyntaxFault
<b>[Reason]</b>	Invalid syntax used for Put request
<b>[Detail]</b>	

1205

1206

### 1207 **5. Security**

1208 It is strongly recommended that the communication between services be secured  
1209 using the mechanisms described in **[WS-Security]**.

1210 In order to properly secure messages, the body (even if empty) and all relevant  
1211 headers need to be included in the signature. Specifically, the WS-Addressing header  
1212 blocks and WS-Security timestamp need to be signed along with the body in order to  
1213 "bind" them together and prevent certain types of attacks.

1214 If a requestor is issuing multiple messages to a resource reference, then it is  
1215 recommended that a security context be established using the mechanisms  
1216 described in **[WS-Trust]** and **[WS-SecureConversation]**. It is further  
1217 recommended that if shared secrets are used, message-specific derived keys also be  
1218 used to protect the secret from crypto attacks.

1219 The access control semantics of resource references are out-of-scope of this  
1220 specification and are specific to each resource reference. Similarly, any protection  
1221 mechanisms on resource references independent of transfer (e.g. embedded  
1222 signatures and encryption) are also out-of-scope.

1223

### 1224 **6. Acknowledgements**

1225 This specification has been developed as a result of joint work with many individuals  
1226 and teams, including: Andrew Hatley (IBM), Denis Rachal (HP), Don Box (Microsoft),  
1227 Don Ferguson (IBM), Frank Vosseler (HP), James Martin (Intel Corporation), John  
1228 Shewchuk (Microsoft), Kirill Gavrylyuk (Microsoft), Mohammad Fakhar (IBM), Savas  
1229 Parastatidis (Microsoft), Steve Millet (Microsoft).

### 1230 **7. References**

#### 1231 **[RFC 2119]**

1232 S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC  
1233 2119, Harvard University, March 1997. (See <http://www.ietf.org/rfc/rfc2119.txt>.)

#### 1234 **[SOAP 1.1]**

1235 D. Box, et al, "Simple Object Access Protocol (SOAP) 1.1," May 2000. (See  
1236 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>.)

#### 1237 **[SOAP 1.2]**

- 1238 M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework," June 2003.  
1239 (See <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>.)
- 1240 **[WS-Addressing]**  
1241 W3C Recommendation, "Web Services Addressing 1.0 (WS-Addressing)," May  
1242 2006. (See <http://www.w3.org/2005/08/addressing/>.)
- 1243 **[WSDL 1.1]**  
1244 E. Christensen, et al, "Web Services Description Language (WSDL) 1.1," March  
1245 2001. (See <http://www.w3.org/TR/2001/NOTE-wsdl-20010315/>.)
- 1246 **[WS-I BP 1.1]**  
1247 K. Ballinger, et al, "WS-I Basic Profile Version 1.1", April 2006. (See  
1248 <http://www.ws-i.org/Profiles/BasicProfile-1.1.html> .)
- 1249 **[WS-MetadataExchange]**  
1250 K. Ballinger, et al, "Web Services Metadata Exchange (WS-MetadataExchange)",  
1251 August 2006. (See <http://schemas.xmlsoap.org/ws/2004/09/mex.>)
- 1252 **[WS-Transfer]**  
1253 J. Alexander, et al, "Web Services Transfer (WS-Transfer)", March 2006. (See  
1254 <http://www.w3.org/Submission/WS-Transfer/>.)
- 1255 **[WS-Security]**  
1256 OASIS standard, "Web Services Security: SOAP Message Security 1.0" (See  
1257 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)  
1258 [security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf))
- 1259 **[WS-Trust]**  
1260 S. Anderson, et al, "Web Services Trust Language (WS-Trust)", February 2005.  
1261 (See <http://schemas.xmlsoap.org/ws/2005/02/trust/> )
- 1262 **[WS-SecureConversation]**  
1263 S. Anderson, et al, "Web Services Secure Conversation Language (WS-  
1264 SecureConversation)", February 2005. (See  
1265 <http://schemas.xmlsoap.org/ws/2005/02/sc/> )
- 1266 **[XML Schema, Part 1]**  
1267 H. Thompson, et al, "XML Schema Part 1: Structures," October 2004. (See  
1268 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>.)
- 1269 **[XML Schema, Part 2]**  
1270 P. Biron, et al, "XML Schema Part 2: Datatypes," October 2004. (See  
1271 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>.)
- 1272 **[XPath 1.0]**  
1273 James Clark, et al, "XML Path Language (XPath) Version 1.0," November 1999,  
1274 (See <http://www.w3.org/TR/xpath> )

## 1275 **Appendix I – XPath Level 1**

1276 XPath Level 1 is a subset of the abbreviated relative syntax of XPath 1.0, and is used  
1277 to identify or select a node within a resource representation or fragment. It is  
1278 identified by the following URI:

1279 `http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/Dialect/XPath-`  
1280 `Level-1`

1281 The following XPath Level 1 grammar is LL(1), and the nonterminal productions are  
1282 in angle brackets. Terminal symbols are either literals, or in UPPERCASE:

1283 **(01)** `<xpath> ::= <context> <node_sequence>;`

1284 **(02)**

1285 **(03)** `<context> ::= '/' | <>;`

```

1286 (04)
1287 (05) <node_sequence> ::=
1288 (06)   <element> <optional_collection_operator> <more>;
1289 (07)
1290 (08) <optional_collection_operator> ::= '[' <array_location> ']';
1291 (09) <optional_collection_operator> ::= <>;
1292 (10)
1293 (11) <more> ::= '/' <follower> | <>;
1294 (12)
1295 (13) <follower> ::=
1296 (14)   <attribute> | <text_function> | <node_sequence>;
1297 (15)
1298 (16) <element>           ::= <qualified_name>;
1299 (17) <attribute>         ::= '@' <qualified_name>;
1300 (18)
1301 (19) <qualified_name> ::= <name> <qname_follower>;
1302 (20) <qname_follower> ::= ':' <name> | <>;
1303 (21) <text_function>   ::= "text()" ;
1304 (22) <array_location> ::= NONZERO_DECIMAL_UNSIGNED_INTEGER;
1305 (23) <name>           ::= XML_TOKEN;

```

1306 The terminal tokens which require further lexical specification are  
1307 NONZERO\_DECIMAL\_UNSIGNED\_INTEGER, whose values are in the subrange (1..  
1308 4294967295), and XML\_TOKEN whose values are equivalent to those for the XML  
1309 Schema type *xs:token*. This grammar is small enough that it can be easily  
1310 implemented in resource-constrained implementations.

1311 The following comments on the grammar will clarify certain constructs within the  
1312 BNF.

1313 Most of the examples assume the following XML sample acting as a "resource"  
1314 document:

```

1315 (01) <a>
1316 (02)   <b>
1317 (03)     <c d="30"> 20 </c>
1318 (04)   </b>
1319 (05)   <e>
1320 (06)     <f/>
1321 (07)     <f/>
1322 (08)   </e>
1323 (09) </a>

```

1324 The context and document root node need clarification. XPath Level 1 assumes that  
1325 the root is the root node of the resource document, not the SOAP envelope or any  
1326 other wrapper element which may contain the resource.

1327 Further, the default context is the root element and the context position is 1.

1328 In view of this, the / operator selects the containing root, and the only valid operand  
1329 which may follow it is the outermost element of the resource:

```

1330 (01) /a

```

1331 The following paths are equivalent:



1332 (01) /a/b  
1333 (02) b

1334 Note that because the context node is the root element, a relative path selects a  
1335 matching child element.

1336 The <node\_sequence> production provides the recursive behavior for the XPath:

1337 (01) /a/b/c  
1338 (02) b/c

1339 It also provides for selecting specific repeated elements through the  
1340 <optional\_collection\_operator> production:

1341 (01) /a/e/f[2]

1342 The collection operator only takes unsigned nonzero values, as defined above for  
1343 NONZERO\_DECIMAL\_UNSIGNED\_INTEGER. Thus, [1] is the first of a repeating  
1344 series of elements.

1345 The <qualified\_name> production allows the XML naming tokens to be either  
1346 namespace-qualified or unqualified:

1347 (01) /ns1:a/ns2:b/c

1348 The namespace bindings are evaluated against any namespace declarations that are  
1349 in scope where the XPath appears within the SOAP message.

1350 NOTE: If the element name is unqualified, i.e. appears without a namespace prefix,  
1351 then the element name MUST be matched against a matching element name in the  
1352 resource document, regardless of namespace bindings that are in effect, including  
1353 default bindings. This allows implementations to simply match element names in the  
1354 majority of cases. If namespace bindings are significant for all elements, then  
1355 qualified names must be used.

1356 The <follower> production allows for special-casing of the final tokens of the XPath  
1357 allowing it to end in either an attribute or text.

1358

1359 The text() NodeTest may be applied as a final token to the selected element. This  
1360 NodeTest selects any text nodes that are children of the selected element. If the  
1361 element only contains text content, the return value will be a node-set containing a  
1362 single text node.

1363 (01) b/c/text()

1364 The above expression would return a node-set containing a single text node with the  
1365 value 20 as its result. This text node would then be serialized into the following XML  
1366 representation:

1367 (01) <wsrt:TextNode>20</wsrt:TextNode>

1368

1369 If accessed, attributes must be the final token in the path and they may be  
1370 namespace-qualified or unqualified names, as required:

1371 (01) /a/b/c/@d

1372 The above expression would return a node-set containing a single attribute node with  
1373 the value d="30" as its result. This attribute node would then be serialized into the  
1374 following XML representation:

1375 (01) <wsrt:AttributeNode name="d">30</wsrt:AttributeNode>

1376

1377 Selection of an element returns the element and its entire content. The path `/a/b`  
1378 executed against the sample XML returns a node-set containing a single element  
1379 node which serializes directly:

```
1380 (01) <b> <c d="30"> 20 </c> </b>
```

1381

1382 In the event that there is more than one node which would match the XPath, the  
1383 implementation SHOULD select or return the first node only. This allows simple  
1384 implementations to avoid the overhead of checking the remainder of the resource  
1385 document for a possible match.

1386 Conformant implementations MAY supply additional functions and capabilities, but  
1387 MUST adhere to the minimum behavior described above.

1388

## 1389 Appendix II – Resource Metadata Content

1390 A resource can have associated metadata that MAY be specified when the resource is  
1391 created. A resource may provide access to that metadata after it has been created  
1392 and some aspects of a resource's metadata may be mutable.

1393

1394 The form of the resource metadata is shown in Table 15.

1395 **Table 15: Resource metadata**

```
1396 (01) <wsrt:Metadata>  
1397 (02)   <wsrt:Lifetime>lifetime metadata</wsrt:Lifetime> ?  
1398 (03)   <wsrt:SupportedDialect>  
1399 (04)     dialect metadata  
1400 (05)   </wsrt:SupportedDialect> *  
1401 (06)   ...  
1402 (07) </wsrt:Metadata>
```

1403

1404 Metadata can be associated with a resource as described in **WS-**  
1405 **MetadataExchange**. The following `wsmex:GetMetadata/wsmex:Dialect` URI is  
1406 defined to indicate metadata as defined in this specification:

```
1407 http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
```

1408 This is used in the `wsmex:GetMetadata` message to return resource metadata. It is  
1409 RECOMMENDED that a resource whose metadata is mutable use the form of a  
1410 `wsmex:MetadataSection` that contains an EPR which is a reference to a "metadata  
1411 resource".

1412 The metadata defined by this specification includes lifecycle metadata as well as  
1413 capability information about supported dialects as described in the following sub-  
1414 sections.

1415

### 1416 II.A Lifecycle metadata

1417 Resources have a distinct lifecycle in that they may be created and they may be  
1418 destroyed. There is no distinction between a resource that has been destroyed and a  
1419 resource that has not been created.

1420 Resources MAY allow their lifecycle metadata to be queried and changed and MAY  
1421 support operations to operate on their lifecycle metadata. The following are  
1422 properties of a resource's lifecycle metadata:

1423 [termination time]  
1424 The time at which the resource will be destroyed. The environment controlling  
1425 the resource MUST NOT destroy the resource before this time but MAY choose to  
1426 delay its destruction after this time. However, client applications MUST NOT  
1427 assume that the resource will be available beyond this date/time.

1428 [current time]  
1429 The current time, as measured by the resource. This can be used to estimate  
1430 local clock variations between time measured by a resource and time measured  
1431 by an application that uses the resource.

1432 The form of resource lifecycle metadata is shown in Table 16.

1433 **Table 16: Resource lifecycle metadata**

```
1434 (01) <wsrt:Lifetime>  
1435 (02)   ( <wsrt:TerminateAt>  
1436 (03)     <wsrt:TerminationTime>xs:dateTime</wsrt:TerminationTime>  
1437 (04)     <wsrt:CurrentTime>xs:dateTime</wsrt:CurrentTime>  
1438 (05)   </wsrt:TerminateAt> |  
1439 (06)   <wsrt:TerminateAfter>xs:duration</wsrt:TerminateAfter> |  
1440 (07)   <wsrt:TerminateAfterIdle>  
1441 (08)     xs:duration  
1442 (09)   </wsrt:TerminateAfterIdle> )  
1443 (10) </wsrt:Lifetime>
```

1444 `wsrt:Lifetime/wsrt:TerminateAt`  
1445 This element contains elements that specify the [termination time] and [current  
1446 time] as absolute times, as measured by the resource.

1447 `wsrt:Lifetime/wsrt:TerminateAt/wsrt:TerminationTime`  
1448 This element specifies the [termination time] as an absolute time, as measured  
1449 by the resource, after which the resource will be destroyed.

1450 `wsrt:Lifetime/wsrt:TerminateAt/wsrt:CurrentTime`  
1451 This element specifies the [current time] as an absolute time, as measured by  
1452 the resource. This can be used to estimate local clock variations between time  
1453 measured by a resource and time measured by an application that uses the  
1454 resource.

1455 `wsrt:Lifetime/wsrt:TerminateAfter`  
1456 This element specifies the [termination time] as a duration after the current time  
1457 that the resource will be destroyed.

1458 `wsrt:Lifetime/wsrt:TerminateAfterIdle`  
1459 This element specifies the [termination time] as an amount of time to wait after a  
1460 message to the resource before automatically destroying it. Any message sent to  
1461 the resource SHOULD reset this timer.

## 1462 II.B Expression Dialect metadata

1463 Resources can support different expression dialects, as described above in  
1464 Expression Dialect. A resource MAY declare which dialects it supports through its  
1465 resource metadata.

1466 The form of resource expression dialect metadata is shown in Table 17.

1467 **Table 17: Resource expression metadata**

```
1468 (01) <wsrt:SupportedDialect DialectName="xs:anyURI">  
1469 (02) <wsrt:SupportedOperation OperationName="xs:anyURI">
```

1470 (03) <wsrt:SupportedPutMode>  
1471 (04) *ModeType*  
1472 (05) </wsrt:SupportedPutMode> \*  
1473 (06) <wsrt:MultipartLimit>  
1474 (07) *xs:positiveInteger*  
1475 (08) </wsrt:MultipartLimit> ?  
1476 (09) </wsrt:SupportedOperation> \*  
1477 (10) </wsrt:SupportedDialect>

1478 wsrt:SupportedDialect  
1479 This element encapsulates all the metadata about the support of a specific  
1480 expression dialect. The resource MUST support each of the dialects in this list and  
1481 MAY support others.

1482 wsrt:SupportedDialect/@DialectName  
1483 The URI that uniquely identifies the dialect.

1484 wsrt:SupportedDialect/wsrt:SupportedOperation  
1485 This element encapsulates all the metadata regarding the behaviour of the  
1486 subject expression dialect for a specific WS-Transfer operation. If this element is  
1487 absent, then all WS-Transfer operations and all Put Modes are supported for the  
1488 subject dialect.

1489 wsrt:SupportedDialect/wsrt:SupportedOperation/@OperationName  
1490 The Action URI that indicates the Get, Put or Create operation.

1491 wsrt:SupportedDialect/wsrt:SupportedOperation/wsrt:SupportedPutMode  
1492 This element contains a PutMode that is supported for the operation for the  
1493 subject dialect. If this element is absent, then all Put Modes are supported for the  
1494 operation for the subject dialect. This element is present only when the  
1495 @OperationName indicates the "Put" operation.

1496 wsrt:SupportedDialect/wsrt:SupportedOperation/wsrt:MultipartLimit  
1497 Indicates the maximum number of <wsrt:Expression> elements supported for a  
1498 Get operation or the maximum number of <wsrt:Fragment> elements supported  
1499 for a Put or Create operation for the subject dialect. If this element is absent then  
1500 there is no limit for the operation for the subject dialect. A resource that specifies  
1501 this metadata MUST generate a MultipartLimitExceededFault if it receives a  
1502 message that exceeds the limit of wsrt:Expression or wsrt:Fragment elements  
1503 supported for the operation for the subject dialect.

## 1504 **Appendix III – XML Schema**

1505  
1506 A normative copy of the XML Schema [[XML Schema Part 1, Part 2](#)] description for  
1507 this specification can be retrieved from the following address:

1508 <http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/wsrt.xsd>

1509 A non-normative copy of the XML Schema description is listed below for convenience.

```
1510 <?xml version="1.0" ?>
1511 <!--
1512 Copyright Notice
1513 (c) 2006 Hewlett-Packard Development Company (HP), Intel Corporation,
1514 International Business Machines Corporation (IBM), and Microsoft
1515 Corporation. All rights reserved.
```

1516

1517 Permission to copy and display the "Web Services Resource Transfer"  
1518 Specification, in any medium without fee or royalty is hereby granted,  
1519 provided that you include the following on ALL copies of the "Web  
1520 Services Resource Transfer" Specification, or portions thereof, that  
1521 you make:

- 1522 1. A link or URL to the "Web Services Resource Transfer"  
1523 Specification at this location:  
1524 <http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer>.
- 1525 2. The copyright notice as shown in the "Web Services Resource  
1526 Transfer" Specification.

1527  
1528 Hewlett-Packard Development Company (HP), Intel Corporation,  
1529 International Business Machines Corporation (IBM), and Microsoft  
1530 Corporation (collectively, the "Authors") each agree to grant you a  
1531 royalty-free license, under reasonable, non-discriminatory terms and  
1532 conditions to their respective patents that they deem necessary to  
1533 implement the "Web Services Resource Transfer" Specification.

1534  
1535 THE "WEB SERVICES RESOURCE TRANSFER" SPECIFICATION IS PROVIDED "AS IS,"  
1536 AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR  
1537 IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,  
1538 FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE  
1539 CONTENTS OF THE "WEB SERVICES RESOURCE TRANSFER" SPECIFICATION ARE  
1540 SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS  
1541 WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR  
1542 OTHER RIGHTS.

1543  
1544 THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,  
1545 INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY  
1546 USE OR DISTRIBUTION OF THE "WEB SERVICES RESOURCE TRANSFER"  
1547 SPECIFICATION.

1548  
1549 The name and trademarks of the Authors may NOT be used in any manner,  
1550 including advertising or publicity pertaining to the "Web Services  
1551 Resource Transfer" Specification or its contents without specific,  
1552 written prior permission. Title to copyright in the "Web Services  
1553 Resource Transfer" Specification will at all times remain with the  
1554 Authors.

1555  
1556 No other rights are granted by implication, estoppel or otherwise.

1557 -->

1558

1559 <xs:schema

1560 targetNamespace="http://schemas.xmlsoap.org/ws/2006/08/resourceTr  
1561 ansfer"

```

1562 xmlns:wsrt="http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer"
1563     xmlns:xs="http://www.w3.org/2001/XMLSchema"
1564     xmlns:wsa="http://www.w3.org/2005/08/addressing"
1565     xmlns:wsmex="http://schemas.xmlsoap.org/ws/2004/09/mex"
1566     elementFormDefault="qualified" blockDefault="#all">
1567
1568     <xs:import
1569         namespace="http://www.w3.org/2005/08/addressing"
1570         schemaLocation=
1571             "http://www.w3.org/2006/03/addressing/ws-addr.xsd" />
1572     <xs:import
1573         namespace="http://schemas.xmlsoap.org/ws/2004/09/mex"
1574         schemaLocation=
1575     "http://schemas.xmlsoap.org/ws/2004/09/mex/MetadataExchange.xsd" />
1576
1577
1578     <!-- ResourceMetadata section -->
1579
1580     <!-- A wsmex:MetadaSection with a wsmex:Dialect of
1581         http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
1582         contains the following Metadata element
1583     -->
1584     <xs:element name="Metadata">
1585         <xs:complexType>
1586             <xs:sequence>
1587                 <xs:element ref="wsrt:Lifetime" minOccurs="0" />
1588                 <xs:element ref="wsrt:SupportedDialect"
1589                     minOccurs="0" maxOccurs="unbounded" />
1590                 <xs:any minOccurs="0" maxOccurs="unbounded"
1591                     namespace="##other" processContents="lax" />
1592             </xs:sequence>
1593         </xs:complexType>
1594     </xs:element>
1595
1596     <xs:element name="Lifetime">
1597         <xs:complexType>
1598             <xs:sequence>
1599                 <xs:choice>
1600                     <xs:element ref="wsrt:TerminateAt"/>
1601                     <xs:element name="TerminateAfter" type="xs:duration" />
1602                     <xs:element name="TerminateAfterIdle"
1603                         type="xs:duration" />
1604                 </xs:choice>
1605             </xs:sequence>
1606         </xs:complexType>

```

```

1607     </xs:element>
1608
1609     <xs:element name="TerminateAt">
1610         <xs:complexType>
1611             <xs:sequence>
1612                 <xs:element name="TerminationTime" type="xs:dateTime"/>
1613                 <xs:element name="CurrentTime" type="xs:dateTime" />
1614             </xs:sequence>
1615         </xs:complexType>
1616     </xs:element>
1617
1618     <xs:element name="SupportedDialect">
1619         <xs:complexType>
1620             <xs:sequence>
1621                 <xs:element ref="wsrt:SupportedOperation"
1622                     minOccurs="0" maxOccurs="unbounded" />
1623             </xs:sequence>
1624             <xs:attribute name='DialectName' type='xs:anyURI'
1625                 use='required' />
1626         </xs:complexType>
1627     </xs:element>
1628
1629     <xs:element name="SupportedOperation">
1630         <xs:complexType>
1631             <xs:sequence>
1632                 <xs:element name="SupportedPutMode" type="wsrt:ModeType"
1633                     minOccurs="0" maxOccurs="unbounded" />
1634                 <xs:element name="MultipartLimit"
1635                     type="xs:positiveInteger" minOccurs="0"/>
1636             </xs:sequence>
1637             <xs:attribute name='OperationName' type='xs:anyURI'
1638                 use='required' />
1639         </xs:complexType>
1640     </xs:element>
1641
1642     <!-- Shared Types section -->
1643     <xs:complexType name='MixedAnyType' final='restriction'
1644         mixed="true">
1645         <xs:complexContent mixed="true">
1646             <xs:restriction base="xs:anyType">
1647                 <xs:sequence>
1648                     <xs:any processContents="lax"
1649                         minOccurs="0" maxOccurs="unbounded" />
1650                 </xs:sequence>
1651             </xs:restriction>

```

```

1652     </xs:complexContent>
1653 </xs:complexType>
1654
1655 <xs:complexType name='ResultType' final='restriction'>
1656     <xs:complexContent>
1657         <xs:extension base='wsrt:MixedAnyType' />
1658     </xs:complexContent>
1659 </xs:complexType>
1660
1661 <xs:complexType name='ExpressionType' final='restriction'>
1662     <xs:complexContent>
1663         <xs:extension base='wsrt:MixedAnyType' />
1664     </xs:complexContent>
1665 </xs:complexType>
1666 <xs:element name='Expression' type='wsrt:ExpressionType' />
1667
1668 <xs:complexType name='FragmentType'>
1669     <xs:sequence>
1670         <xs:element ref='wsrt:Expression'
1671             minOccurs='0' maxOccurs='1' />
1672         <xs:element name='Value' type='wsrt:MixedAnyType'
1673             minOccurs='0' maxOccurs='1' />
1674     </xs:sequence>
1675 </xs:complexType>
1676 <xs:element name='Fragment' type='wsrt:FragmentType' />
1677
1678 <xs:element name="ResourceTransfer">
1679     <xs:complexType>
1680         <xs:anyAttribute namespace="##other" processContents="lax" />
1681     </xs:complexType>
1682 </xs:element>
1683
1684 <!-- Create section -->
1685
1686 <xs:element name='Create'>
1687     <xs:complexType>
1688         <xs:sequence>
1689             <xs:element ref='wsmex:Metadata' minOccurs='0' />
1690             <xs:element ref='wsrt:Fragment'
1691                 minOccurs='0' maxOccurs='unbounded' />
1692         </xs:sequence>
1693         <xs:attribute name='Dialect' type='xs:anyURI' />
1694     </xs:complexType>
1695 </xs:element>
1696

```



```

1697 <xs:element name="CreateResponse">
1698   <xs:complexType>
1699     <xs:sequence>
1700       <xs:element ref="wsrt:ResourceCreated" />
1701     </xs:sequence>
1702   </xs:complexType>
1703 </xs:element>
1704
1705 <xs:element name="ResourceCreated"
1706   type="wsa:EndpointReferenceType" />
1707
1708 <!-- Put section -->
1709
1710 <xs:element name='Put'>
1711   <xs:complexType>
1712     <xs:sequence>
1713       <xs:element name='Fragment' type='wsrt:PutFragmentType'
1714         maxOccurs='unbounded' />
1715     </xs:sequence>
1716     <xs:attribute name='Dialect' type='xs:anyURI' />
1717   </xs:complexType>
1718 </xs:element>
1719
1720 <xs:complexType name='PutFragmentType'>
1721   <xs:complexContent>
1722     <xs:extension base="wsrt:FragmentType">
1723       <xs:attribute name='Mode' type='wsrt:ModeType'
1724         use='required' />
1725     </xs:extension>
1726   </xs:complexContent>
1727 </xs:complexType>
1728
1729 <xs:simpleType name="ModeType" final="restriction">
1730   <xs:restriction base="xs:NMTOKEN">
1731     <xs:enumeration value="Modify" />
1732     <xs:enumeration value="Insert" />
1733     <xs:enumeration value="Remove" />
1734   </xs:restriction>
1735 </xs:simpleType>
1736
1737 <xs:element name="PutResponse">
1738   <xs:complexType/>
1739 </xs:element>
1740
1741 <!-- Get section -->

```

```

1742
1743     <xs:element name='Get'>
1744         <xs:complexType>
1745             <xs:sequence>
1746                 <xs:element ref='wsrt:Expression'
1747                     minOccurs='0' maxOccurs='unbounded' />
1748             </xs:sequence>
1749             <xs:attribute name='Dialect' type='xs:anyURI' />
1750         </xs:complexType>
1751     </xs:element>
1752
1753     <xs:element name='GetResponse'>
1754         <xs:complexType>
1755             <xs:sequence>
1756                 <xs:element name='Result' type='wsrt:ResultType'
1757                     maxOccurs='unbounded' />
1758             </xs:sequence>
1759         </xs:complexType>
1760     </xs:element>
1761
1762     <!-- Fault section -->
1763
1764     <xs:simpleType name="FaultCodeTypes">
1765         <xs:restriction base="xs:QName">
1766             <xs:enumeration value="wsrt:ConcurrencyFault" />
1767             <xs:enumeration value="wsrt:UnsupportedDialectFault" />
1768             <xs:enumeration value="wsrt:InvalidExpressionFault" />
1769             <xs:enumeration value="wsrt:GetFault" />
1770             <xs:enumeration value="wsrt:ResourceValidityFault" />
1771             <xs:enumeration value="wsrt:FragmentAlreadyExistsFault" />
1772             <xs:enumeration value="wsrt:PutFault" />
1773             <xs:enumeration value="wsrt:PutModeUnsupportedFault" />
1774             <xs:enumeration value="wsrt:CreateFault" />
1775             <xs:enumeration value="wsrt:InvalidMetadataFault" />
1776             <xs:enumeration value="wsrt:MultipartLimitExceededFault" />
1777             <xs:enumeration value="wsrt:InvalidPutSyntaxFault" />
1778         </xs:restriction>
1779     </xs:simpleType>
1780
1781     <xs:element name='Dialect' type='xs:anyURI' />
1782
1783     <xs:element name='InvalidExpressionSyntax'>
1784         <xs:complexType>
1785             <xs:sequence>
1786                 <xs:element ref='wsrt:Expression'

```

```

1787         maxOccurs='unbounded' />
1788     </xs:sequence>
1789 </xs:complexType>
1790 </xs:element>
1791
1792 <xs:element name='InvalidExpressionValue'>
1793     <xs:complexType>
1794         <xs:sequence>
1795             <xs:element ref='wsrt:Expression'
1796                 maxOccurs='unbounded' />
1797         </xs:sequence>
1798     </xs:complexType>
1799 </xs:element>
1800
1801 <xs:element name='MultipartLimit' type='xs:positiveInteger' />
1802
1803 <xs:element name='SideEffects' type='xs:boolean' />
1804
1805 <!-- XPath section -->
1806
1807 <xs:element name='AttributeNode'>
1808     <xs:complexType>
1809         <xs:simpleContent>
1810             <xs:extension base='xs:string'>
1811                 <xs:attribute name='name' type='xs:QName' />
1812             </xs:extension>
1813         </xs:simpleContent>
1814     </xs:complexType>
1815 </xs:element>
1816
1817
1818 <xs:element name='TextNode' type='xs:string' />
1819
1820 <xs:element name='NodeSet' type='wsrt:ResultType' />
1821
1822 </xs:schema>

```

## 1823 **Appendix IV – WSDL**

1824 A normative copy of the WSDL [[WSDL 1.1](#)] description for this specification can be  
1825 retrieved from the following address:

1826 <http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/wsrt.wsdl>

1827 A non-normative copy of the WSDL description is listed below for convenience.

```

1828 <?xml version="1.0" encoding="utf-8"?>
1829 <!--
1830 Copyright Notice

```

1831 (c) 2006 Hewlett-Packard Development Company (HP), Intel Corporation,  
1832 International Business Machines Corporation (IBM), and Microsoft  
1833 Corporation. All rights reserved.

1834

1835 Permission to copy and display the "Web Services Resource Transfer"  
1836 Specification, in any medium without fee or royalty is hereby granted,  
1837 provided that you include the following on ALL copies of the "Web  
1838 Services Resource Transfer" Specification, or portions thereof, that  
1839 you make:

1840 1. A link or URL to the "Web Services Resource Transfer"  
1841 Specification at this location:

1842 <http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer>.

1843 2. The copyright notice as shown in the "Web Services Resource  
1844 Transfer" Specification.

1845

1846 Hewlett-Packard Development Company (HP), Intel Corporation,  
1847 International Business Machines Corporation (IBM), and Microsoft  
1848 Corporation (collectively, the "Authors") each agree to grant you a  
1849 royalty-free license, under reasonable, non-discriminatory terms and  
1850 conditions to their respective patents that they deem necessary to  
1851 implement the "Web Services Resource Transfer" Specification.

1852

1853 THE "WEB SERVICES RESOURCE TRANSFER" SPECIFICATION IS PROVIDED "AS IS,"  
1854 AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR  
1855 IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY,  
1856 FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE  
1857 CONTENTS OF THE "WEB SERVICES RESOURCE TRANSFER" SPECIFICATION ARE  
1858 SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS  
1859 WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR  
1860 OTHER RIGHTS.

1861

1862 THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL,  
1863 INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY  
1864 USE OR DISTRIBUTION OF THE "WEB SERVICES RESOURCE TRANSFER"  
1865 SPECIFICATION.

1866

1867 The name and trademarks of the Authors may NOT be used in any manner,  
1868 including advertising or publicity pertaining to the "Web Services  
1869 Resource Transfer" Specification or its contents without specific,  
1870 written prior permission. Title to copyright in the "Web Services  
1871 Resource Transfer" Specification will at all times remain with the  
1872 Authors.

1873

1874 No other rights are granted by implication, estoppel or otherwise.

1875 -->

```
1876
1877 <wsdl:definitions targetNamespace=
1878     "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer"
1879     xmlns:wsrt="http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer"
1880     xmlns:wsa="http://www.w3.org/2005/08/addressing"
1881     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1882     xmlns:wxf="http://schemas.xmlsoap.org/ws/2004/09/transfer"
1883     xmlns:xs="http://www.w3.org/2001/XMLSchema">
1884
1885     <wsdl:types>
1886         <xs:schema>
1887             <xs:include schemaLocation=
1888 "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/wsrt.xsd" />
1889             </xs:schema>
1890         </wsdl:types>
1891
1892     <wsdl:message name="CreateRequestMessage">
1893         <wsdl:part name="Body" element="wsrt:Create" />
1894     </wsdl:message>
1895
1896     <wsdl:message name="CreateResponseMessage">
1897         <wsdl:part name="Body" element="wsrt:CreateResponse" />
1898     </wsdl:message>
1899
1900     <wsdl:message name="GetRequestMessage">
1901         <wsdl:part name="Body" element="wsrt:Get" />
1902     </wsdl:message>
1903
1904     <wsdl:message name="GetResponseMessage">
1905         <wsdl:part name="Body" element="wsrt:GetResponse" />
1906     </wsdl:message>
1907
1908     <wsdl:message name="PutRequestMessage">
1909         <wsdl:part name="Body" element="wsrt:Put" />
1910     </wsdl:message>
1911
1912     <wsdl:message name="PutResponseMessage">
1913         <wsdl:part name="Body" element="wsrt:PutResponse" />
1914     </wsdl:message>
1915
1916     <wsdl:portType name="ResourceInterface">
1917         <wsdl:documentation>
1918             This port type contains the Get and Put
1919             operations defined in WS-ResourceTransfer.
1920         </wsdl:documentation>
```

```
1921     <wsdl:operation name="Get">
1922         <wsdl:input message="wsrt:GetRequestMessage"
1923             wsa:Action=
1924                 "http://schemas.xmlsoap.org/ws/2004/09/transfer/Get" />
1925         <wsdl:output message="wsrt:GetResponseMessage"
1926             wsa:Action=
1927                 "http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse" />
1928     </wsdl:operation>
1929     <wsdl:operation name="Put">
1930         <wsdl:input message="wsrt:PutRequestMessage"
1931             wsa:Action=
1932                 "http://schemas.xmlsoap.org/ws/2004/09/transfer/Put" />
1933         <wsdl:output message="wsrt:PutResponseMessage"
1934             wsa:Action=
1935                 "http://schemas.xmlsoap.org/ws/2004/09/transfer/PutResponse" />
1936     </wsdl:operation>
1937 </wsdl:portType>
1938
1939 <wsdl:portType name="ResourceFactoryInterface">
1940     <wsdl:documentation>
1941         This port type contains the Create operation
1942         defined in WS-ResourceTransfer.
1943     </wsdl:documentation>
1944     <wsdl:operation name="Create">
1945         <wsdl:input message="wsrt:CreateRequestMessage"
1946             wsa:Action=
1947                 "http://schemas.xmlsoap.org/ws/2004/09/transfer/Create" />
1948         <wsdl:output message="wsrt:CreateResponseMessage"
1949             wsa:Action=
1950                 "http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse" />
1951     </wsdl:operation>
1952 </wsdl:portType>
1953
1954 </wsdl:definitions>
1955
```