Web Services Resource Transfer (WS-RT)

Version 1.0, August 2006

Authors

Brian Reistad, Microsoft Corporation
Bryan Murray, HP
Doug Davis, IBM
Ian Robinson (Editor), IBM
Raymond McCollum (Editor), Microsoft Corporation
Alexander Nosov, Microsoft Corporation
Steve Graham, IBM
Vijay Tewari, Intel Corporation
William Vambenepe, HP

Copyright Notice

Abstract

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

Composable Architecture

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

Status

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

Table of Contents

1. Introduction

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

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

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

1.1 Requirements

This specification intends to meet the following requirements:

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

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

1.2 Non-Requirements

This specification does not intend to meet the following requirements:

- Discovery of resources

1.3 Example

This section contains a complete example of a WS-RT "Get" operation. This example is meant for illustration only and does not represent normative behavior or content.

Table 1 shows the XML representation of the example resource which will be accessed by the protocol operation. The example resource is a physical disk which has a number of logical volumes.

Table 1: Example resource

```
(01)<Disk xmlns="http://example.org/sample">
(02)    <DiskCapacity>62500000000</DiskCapacity>
(03)    <DiskFreeSpace>524182841</DiskFreeSpace>
(04)    <SerialNumber>123-F2560</SerialNumber>
(05)    <LastAuditDate>1998-05-25T13:30:15</LastAuditDate>
(06)    <Volume>
(07)       <Drive>C:</Drive>
(08)       <Label>MyDrive-C</Label>
(09)       <TotalCapacity>10000000000</TotalCapacity>
(10)       <FreeSpace>6234794528</FreeSpace>
(11)    </Volume>
(12)    <Volume>
(13)       <Drive>D:</Drive>
(14)       <Label>MyDrive-D</Label>
(15)       <TotalCapacity>30000000000</TotalCapacity>
(16)       <FreeSpace>26462809800</FreeSpace>
(17)    </Volume>
(18)    <Volume>
(19)       <Drive>E:</Drive>
(20)       <Label>MyDrive-E</Label>
(21)       <TotalCapacity>22500000000</TotalCapacity>
(22)       <FreeSpace>16056784170</FreeSpace>
(23)    </Volume>
(24) </Disk>
```

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

Table 2: Example "Get" operation of resource content

```
(01)<s:Envelope
(02)    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
(03)    xmlns:wsa="http://www.w3.org/2005/08/addressing"
(04)    xmlns:wsrt=
(05)       "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer">
(06) <s:Header>
(07)    <wsa:To>http://www.example.org/disk</wsa:To>
(08)    <wsa:Action s:mustUnderstand="true">
(09)      http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(10)    </wsa:Action>
(11)    <wsrt:ResourceTransfer s:mustUnderstand="true"/>
(12)    ...
(13)</s:Header>
(14) <s:Body>
(15)    <wsrt:Get
(16)      Dialect="http://schemas.xmlsoap.org/ws/2006/08/
(17)                  resourceTransfer/Dialect/XPath-Level-1"
(18)           xmlns:d="http://example.org/sample">
(19)       <wsrt:Expression>
(20)           d:Volume[1]/d:Label
(21)       </wsrt:Expression>
(22)       <wsrt:Expression>
(23)           d:DiskCapacity
(24)       </wsrt:Expression>
(25)       <wsrt:Expression>
(26)           d:SerialNumber/text()
(27)       </wsrt:Expression>
(28)    </wsrt:Get>
(29) </s:Body>
(30)</s:Envelope>
```

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

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

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

Table 3: Example "Get" response.

```
(01)<s:Envelope
(02)    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
(03)    xmlns:wsa="http://www.w3.org/2005/08/addressing"
(04)    xmlns:wsrt=
(05)       "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer">
(06) <s:Header>
(07)    <wsa:To>
(08)  http://www.w3.org/2005/08/addressing/anonymous
(09)    </wsa:To>
(10)    <wsa:Action s:mustUnderstand="true">
(11) http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
(12)    </wsa:Action>
(13)    <wsrt:ResourceTransfer/>
(14)    ...
(15) </s:Header>
(16) <s:Body>
(17)  <wsrt:GetResponse xmlns:d="http://example.org/sample">
(18)     <wsrt:Result>
(19)       <d:Label>MyDrive-C</d:Label>
(20)     </wsrt:Result>
(21)     <wsrt:Result>
(22)       <d:DiskCapacity>62500000000</d:DiskCapacity>
(23)     </wsrt:Result>
(24)     <wsrt:Result>
(25)       <wsrt:TextNode>123-F2560</wsrt:TextNode>
(26)     </wsrt:Result>
(27)  </wsrt:GetResponse>
(28) </s:Body>
(29)</s:Envelope>
```

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

2. Terminology and Notation

2.1 Terminology

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

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

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

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

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

Fragment
: The term “fragment” is used in this specification to mean a part of the resource representation.

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

2.2 XML Namespaces

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

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

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

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	http://schemas.xmlsoap.org/soap/envelope/	[SOAP 1.1]
s12	http://www.w3.org/2003/05/soap-envelope	[SOAP 1.2]
wsa	http://www.w3.org/2005/08/addressing	[WS-Addressing]
wsmex	http://schemas.xmlsoap.org/ws/2004/09/mex	[WS-MetadataExchange]
wsdl	http://schemas.xmlsoap.org/wsdl/	[WSDL 1.1]

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

2.3 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC 2119].

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

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

In addition to Message Information Header properties [WS-Addressing], this specification uses the following properties to define messages:

[Headers]
Unordered message headers.

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

[Body]
A message body.

These properties bind to a SOAP Envelope as follows:

```
<s:Envelope>
  <s:Header>
    [Headers]
    <wsa:Action>[Action]</wsa:Action>
    ...
  </s:Header>
  <s:Body>[Body]</s:Body>
</s:Envelope>
```

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

2.4 Compliance

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

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

3. Extensions to WS-Transfer

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

3.1 Fragments

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

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

3.2 Expression Dialect

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

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

3.2.1 QName Dialect

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

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

The QName dialect MUST be indicated by using the URI:

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

Note that the expression MUST evaluate to zero or more entire elements, each including the element name, any attributes and its entire content. The QName dialect does not support computed values.

3.2.2 XPath Level 1 Dialect

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

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

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

- Context Node: the root element of the XML representation of the resource
- Context Position: 1
- Context Size: 1
- Variable Binding: None
- Node Tests: NameTest and the text NodeType
- Function Libraries: None
- Namespace Declarations: Any namespace declarations in-scope where the XPath expression appears

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

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

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

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

3.2.3 XPath 1.0 Dialect

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

An XPath 1.0 expression is an expression whose context is:

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

- Context Position: 1
- Context Size: 1
- Variable Binding: None
- Function Libraries: Core function library
- Namespace Declarations: Any namespace declarations in-scope where the XPath expression appears

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

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

```
http://www.w3.org/TR/1999/REC-xpath-19991116
```

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

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

- Boolean – converted to an xs:boolean
- string – convert to an xs:string
- number – convert to an xs:double

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

Attribute nodes in XPath are represented in the following form:

```
name="value"
```

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

```
(01) <wsrt:AttributeNode name="attribute name">
(02)     attribute value
(03) </wsrt:AttributeNode>
```

The following describes additional constraints on the outline listed above:

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

wsrt:AttributeNode/@name
: This attribute MUST be the name portion of the attribute node.

Text nodes are serialized in the following form:

```
(01) <wsrt:TextNode>
(02)     text value
(03) </wsrt:TextNode>
```

The following describes additional constraints on the outline listed above:

wsrt:TextNode

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

Given the following XML as an example document.

```
(01)<a xmlns="example">
(02)  <b>1</b>
(03)  <c x="y">2</c>
(04)</a>
```

The result of the XPath “/a/b | /a/b/text() | /a/c/@x” would be serialized as the following:

```
(01)<wsrt:Result>
(02)  <b>1</b>
(03)  <wsrt:TextNode>1</wsrt:TextNode>
(04)  <wsrt:AttributeNode name="x">y</wsrt:AttributeNode>
(05)</wsrt:Result>
```

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

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

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

3.3 Get

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

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

The outline for wsrt:Get is:

```
(01)[Headers]
(02)<wsrt:ResourceTransfer s:mustUnderstand="true"? />
(03)[Action]
(04)http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(05)[Body]
(06)<wsrt:Get Dialect=”xs:anyURI”?>
(07)  <wsrt:Expression ...>xs:any</wsrt:Expression> *
(08)</wsrt:Get>
```

The following describes additional constraints on the outline listed above:

[Header]/wsrt:ResourceTransfer

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

[Body]/wsrt:Get

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

[Body]/wsrt:Get/@Dialect

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

[Body]/wsrt:Get/wsrt:Expression

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

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

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

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

```
(01) [Headers]
(02) <wsrt:ResourceTransfer/>
(03) [Action]
(04) http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
(05) [Body]
(06) <wsrt:GetResponse>
(07)   <wsrt:Result...>xs:any</wsrt:Result> +
(08) </wsrt:GetResponse>
```

The following describes additional constraints on the outline listed above:

[Headers]/wsrt:ResourceTransfer

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

[Body]/wsrt:GetResponse

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

[Body]/wsrt:GetResponse/wsrt:Result

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

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

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

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

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

```
(01)<s:Envelope
(02)    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
(03)    xmlns:wsa="http://www.w3.org/2005/08/addressing"
(04)    xmlns:wsrt=
(05)       "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer">
(06) <s:Header>
(07)    <wsa:To>http://www.example.org/disk</wsa:To>
(08)    <wsa:Action s:mustUnderstand="true">
(09)      http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
(10)    </wsa:Action>
(11)    <wsrt:ResourceTransfer s:mustUnderstand="true"/>
(12)    ...
(13) </s:Header>
(14) <s:Body>
(15)    <wsrt:Get Dialect="http://schemas.xmlsoap.org/ws/2006/08/
(16)                         resourceTransfer/Dialect/QName"
(17)      xmlns:d="http://example.org/sample">
(18)       <wsrt:Expression>
(19)           d:Volume
(20)       </wsrt:Expression>
(21)       <wsrt:Expression>
(22)           d:DiskCapacity
(23)       </wsrt:Expression>
(24)    </wsrt:Get>
(25) </s:Body>
(26)</s:Envelope>
```

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

Table 6: Example GetResponse using the "QName" dialect

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

```
(05)       "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer">
(06) <s:Header>
(07)    <wsa:To>
(08)  http://www.w3.org/2005/08/addressing/anonymous
(09)    </wsa:To>
(10)    <wsa:Action s:mustUnderstand="true">
(11) http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
(12)    </wsa:Action>
(13)    <wsrt:ResourceTransfer/>
(14)    ...
(15) </s:Header>
(16) <s:Body>
(17)  <wsrt:GetResponse xmlns:d="http://example.org/sample">
(18)     <wsrt:Result>
(19)       <d:Volume>
(20)          <d:Drive>C:</d:Drive>
(21)          <d:Label>MyDrive-C</d:Label>
(22)          <d:TotalCapacity>10000000000</d:TotalCapacity>
(23)          <d:FreeSpace>6234794528</d:FreeSpace>
(24)       </d:Volume>
(25)       <d:Volume>
(26)          <d:Drive>D:</d:Drive>
(27)          <d:Label>MyDrive-D</d:Label>
(28)          <d:TotalCapacity>30000000000</d:TotalCapacity>
(29)          <d:FreeSpace>26462809800</d:FreeSpace>
(30)       </d:Volume>
(31)       <d:Volume>
(32)          <d:Drive>E:</d:Drive>
(33)          <d:Label>MyDrive-E</d:Label>
(34)          <d:TotalCapacity>22500000000</d:TotalCapacity>
(35)          <d:FreeSpace>16056784170</d:FreeSpace>
(36)       </d:Volume>
(37)     </wsrt:Result>
(38)     <wsrt:Result>
(39)       <d:DiskCapacity>62500000000</d:DiskCapacity>
(40)     </wsrt:Result>
(41)  </wsrt:GetResponse>
(42) </s:Body>
(43)</s:Envelope>
```

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

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

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

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

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

Table 8: Example GetResponse of an XPath 1.0 query expression

```
(01)<s:Body>
(02)   <wsrt:GetResponse>
(03)     <wsrt:Result>2</wsrt:Result>
(04)   </wsrt:GetResponse>
(05)</s:Body>
```

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

3.4 Put

The WS-Transfer “Put” operation is used to update an existing resource representation by providing a replacement XML representation. WS-ResourceTransfer extends the “Put” operation to update an existing resource representation by providing fragments of the XML representation. A resource that can update its full representation MUST also support wxf:Put to update the entire resource representation without using WS-ResourceTransfer extensions.

The extended outline for the “Put” operation is:

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

The following describes additional constraints on the outline listed above:

[Header]/wsrt:ResourceTransfer

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

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

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

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

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

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

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

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

A value of “Insert” indicates that the fragment MUST be added to the resource representation. If the expression targets a repeated element (maxOccurs > 1), the fragment MUST be added at the end. If the expression targets a non-repeated element (maxOccurrs = 1) that already exists, the resource MUST generate a FragmentAlreadyExistsFault. If the expression targets an existing item of a repeated element, the fragment MUST be added before the existing item.

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

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

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

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

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

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

```
(01) [Headers]
(02) <wsrt:ResourceTransfer/>
(03) [Action]
(04) http://schemas.xmlsoap.org/ws/2004/09/transfer/PutResponse
(05) [Body]
```

The following describes additional constraints on the outline listed above:

[Headers]/wsrt:ResourceTransfer
This header indicates that the response contains body content defined in WS-ResourceTransfer.

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

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

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

```
(01) <s:Body>
(02)   <wsrt:Put Dialect="http://schemas.xmlsoap.org/ws/2006/08/
(03)                 resourceTransfer/Dialect/XPath-Level-1"
(04)       xmlns:d="http://example.org/sample">
(05)     <wsrt:Fragment Mode=″Remove″>
(06)       <wsrt:Expression>
(07)         d:Volume[1]
(08)       </wsrt:Expression>
```

```
(09)     </wsrt:Fragment>
(10)     <wsrt:Fragment Mode=”Insert”>
(11)       <wsrt:Expression>
(12)         d:Volume[2]
(13)       </wsrt:Expression>
(14)       <wsrt:Value>
(15)         <d:Volume>
(16)           <d:Drive>X:</d:Drive>
(17)           <d:Label>MyDrive-X</d:Label>
(18)           <d:TotalCapacity>5000000000</d:TotalCapacity>
(19)         </d:Volume>
(20)       </wsrt:Value>
(21)     </wsrt:Fragment>
(22)   </wsrt:Put>
(23)</s:Body>
```

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

Table 10: Updated resource representation

```
(01)<Disk xmlns="http://example.org/sample">
(02)    <DiskCapacity>62500000000</DiskCapacity>
(03)    <DiskFreeSpace>524182841</DiskFreeSpace>
(04)    <SerialNumber>123-F2560</SerialNumber>
(05)    <LastAuditDate>1998-05-25T13:30:15</LastAuditDate>
(06)    <Volume>
(07)       <Drive>D:</Drive>
(08)       <Label>MyDrive-D</Label>
(09)       <TotalCapacity>30000000000</TotalCapacity>
(10)       <FreeSpace>26462809800</FreeSpace>
(11)    </Volume>
(12)    <Volume>
(13)       <Drive>X:</Drive>
(14)       <Label>MyDrive-X</Label>
(15)       <TotalCapacity>5000000000</TotalCapacity>
(16)       <FreeSpace>5000000000</FreeSpace>
(17)    </Volume>
(18)    <Volume>
(19)       <Drive>E:</Drive>
(20)       <Label>MyDrive-E</Label>
(21)       <TotalCapacity>22500000000</TotalCapacity>
(22)       <FreeSpace>16056784170</FreeSpace>
(23)    </Volume>
```

```
(24) </Disk>
```

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

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

Table 11 – Example Put message using the QName dialect

```
(01)<s:Body>
(02)  <wsrt:Put Dialect="http://schemas.xmlsoap.org/ws/2006/08/
(03)                          resourceTransfer/Dialect/QName"
(04)      xmlns:d="http://example.org/sample">
(05)    <wsrt:Fragment Mode=”Modify”>
(06)      <wsrt:Expression>
(07)        d:Volume
(08)      </wsrt:Expression>
(09)      <wsrt:Value>
(10)        <d:Volume>
(11)          <d:Drive>F:</d:Drive>
(12)          <d:Label>MyDrive-F</d:Label>
(13)          <d:TotalCapacity>5000000000</d:TotalCapacity>
(14)        </d:Volume>
(15)        <d:Volume>
(16)          <d:Drive>D:</d:Drive>
(17)          <d:Label>MyDrive-D</d:Label>
(18)          <d:TotalCapacity>30000000000</d:TotalCapacity>
(19)        </d:Volume>
(20)      </wsrt:Value>
(21)    </wsrt:Fragment>
(22)    <wsrt:Fragment Mode=”Insert”>
(23)      <wsrt:Expression>
(24)        d:Volume
(25)      </wsrt:Expression>
(26)      <wsrt:Value>
(27)        <d:Volume>
(28)          <d:Drive>X:</d:Drive>
(29)          <d:Label>MyDrive-X</d:Label>
(30)          <d:TotalCapacity>5000000000</d:TotalCapacity>
(31)        </d:Volume>
(32)      </wsrt:Value>
(33)    </wsrt:Fragment>
(34)  </wsrt:Put>
(35)</s:Body>
```

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

d:Volume elements, identified by the wsrt:Expression contents on line (07). Lines (09) - (20) show the new value for this set of elements to use to replace the old set.

Lines (22) - (33) indicates that a fragment should be inserted into the representation that results from applying the first fragment update. Lines (26) - (32) show the content of the new Volume which is added at the end.

The updated resource representation is illustrated in Table 12.

Table 12: Updated resource representation

```
(01)<Disk xmlns="http://example.org/sample">
(02)   <DiskCapacity>62500000000</DiskCapacity>
(03)   <DiskFreeSpace>524182841</DiskFreeSpace>
(04)   <SerialNumber>123-F2560</SerialNumber>
(05)   <LastAuditDate>1998-05-25T13:30:15</LastAuditDate>
(06)   <Volume>
(07)     <Drive>F:</Drive>
(08)     <Label>MyDrive-F</Label>
(09)     <TotalCapacity>5000000000</TotalCapacity>
(10)     <FreeSpace>5000000000</FreeSpace>
(11)   </Volume>
(12)   <Volume>
(13)    <Drive>D:</Drive>
(14)    <Label>MyDrive-D</Label>
(15)    <TotalCapacity>30000000000</TotalCapacity>
(16)    <FreeSpace>30000000000</FreeSpace>
(17)   </Volume>
(18)   <Volume>
(19)      <Drive>X:</Drive>
(20)      <Label>MyDrive-X</Label>
(21)      <TotalCapacity>5000000000</TotalCapacity>
(22)      <FreeSpace>5000000000</FreeSpace>
(23)   </Volume>
(24)</Disk>
```

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

3.5 Create

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

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

extends the “Create” operation such that any resource metadata MAY be created as part of the creation of the resource.

The extended outline for the “Create” operation is:

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

The following describes additional constraints on the outline listed above:

[Header]/wsrt:ResourceTransfer
: If present and understood, a resource MUST process the [Body] in its entirety and comply with its content. If not present then a resource MUST treat this request as described in WS-Transfer Create [WS-Transfer].

[Body]/wsrt:Create
: An element that specifies the fragments of the resource representation to be initialized during resource creation and optionally any resource metadata that is to be created as part of the creation of the resource. This element MUST be present if the wsrt:ResourceTransfer header is present.

[Body]/wsrt:Create/@Dialect
: This URI indicates which expression dialect will be used to identify the fragment(s) of the resource representation to be initialized during resource creation. This attribute MUST be present when the message contains a wsrt:Expression element.

[Body]/wsrt:Create/wsmex:Metadata
: When present this optional element MUST contain at least one wsmex:MetadataSection. This is resource metadata to be created and initialized during the creation of the resource.

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

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

[Body]/wsrt:Create/Fragment
: This element encompasses a single resource fragment to be initialized during the resource creation. If there are multiple Fragment elements then the resource MUST appear to have been created as though each fragment were processed in the sequence specified in the Create message.

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

[Body]/wsrt:Create/Fragment/Expression

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

[Body]/wsrt:Create/Fragment/Value

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

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

```
(01) [Headers]
(02) <wsrt:ResourceTransfer/>
(03) [Action]
(04) http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse
(05) [Body]
(06) <wxf:ResourceCreated>
(07)     wsa:EndpointReferenceType
(08) </wxf:ResourceCreated>
```

The following describes additional constraints on the outline listed above:

[Headers]/wsrt:ResourceTransfer

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

[Body]/wxf:ResourceCreated

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

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

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

Table 13 - Example Create message using the QName dialect

```
(01) <s:Body>
(02)   <wsrt:Create Dialect="http://schemas.xmlsoap.org/ws/2006/08/
(03)                          resourceTransfer/Dialect/QName"
(04)       xmlns:d="http://example.org/sample">
(05)     <wsmex:Metadata>
```

```
(06)      <wsmex:MetadataSection Dialect="http://schemas.xmlsoap.org/
(07)                          ws/2006/08/resourceTransfer">
(08)        <wsrt:Metadata>
(09)         <wsrt:Lifetime>
(10)          <wsrt:TerminateAt>
(11)           <wsrt:TerminationTime>
(12)            2006-04-11T12:00:00Z
(13)           </wsrt:TerminationTime>
(14)           <wsrt:CurrentTime>
(15)            2006-04-10T10:00:54Z
(16)           </wsrt:CurrentTime>
(17)          </wsrt:TerminateAt>
(18)         </wsrt:Lifetime>
(19)        </wsrt:Metadata>
(20)      </wsmex:MetadataSection>
(21)    </wsmex:Metadata>
(22)    <wsrt:Fragment>
(23)      <wsrt:Expression>
(24)        d:Volume
(25)      </wsrt:Expression>
(26)      <wsrt:Value>
(27)        <d:Volume>
(28)          <d:Drive>C:</d:Drive>
(29)          <d:Label>MyDrive-C</d:Label>
(30)          <d:TotalCapacity>10000000000</d:TotalCapacity>
(31)        </d:Volume>
(32)        <d:Volume>
(33)          <d:Drive>D:</d:Drive>
(34)          <d:Label>MyDrive-D</d:Label>
(35)          <d:TotalCapacity>30000000000</d:TotalCapacity>
(36)        </d:Volume>
(37)      </wsrt:Value>
(38)    </wsrt:Fragment>
(39)  </wsrt:Create>
(40)</s:Body>
```

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

Table 14: Example CreateResponse

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

```
(04)       <wsa:ReferenceParameters>
(05)         <xyz:ManagedResource>44355</xyz:ManagedResource>
(06)       </wsa:ReferenceParameters>
(07)     </wxf:ResourceCreated>
(08)</s:Body>
```

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

4. Faults

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

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

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

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

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] The English language reason element.

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

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

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

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

```
<s12:Envelope>
 <s12:Header>
   <wsa:Action>
      http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
   </wsa:Action>
   <!-- Headers elided for clarity.  -->
 </s12:Header>
 <s12:Body>
  <s12:Fault>
   <s12:Code>
     <s12:Value>[Code]</s12:Value>
     <s12:Subcode>
      <s12:Value>[Subcode]</s12:Value>
     </s12:Subcode>
```

```
   </s12:Code>
   <s12:Reason>
     <s12:Text xml:lang="en">[Reason]</s12:Text>
   </s12:Reason>
   <s12:Detail>
     [Detail]
        ...
   </s12:Detail>
  </s12:Fault>
 </s12:Body>
</s12:Envelope>
```

The properties bind to a SOAP 1.1 fault as follows:

```
<s11:Envelope>
 <s11:Body>
  <s11:Fault>
   <faultcode>[Subcode]</faultcode>
   <faultstring xml:lang="en">[Reason]</faultstring>
   <detail>
     [Detail]
        ...
   </detail>
  </s11:Fault>
 </s11:Body>
</s11:Envelope>
```

4.1 wsa:DestinationUnreachable

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

The SOAP bindings for this fault are defined in [WS-Addressing].

4.2 wsa:EndpointUnavailable

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

The SOAP bindings for this fault are defined in [WS-Addressing].

4.3 ConcurrencyFault

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

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

4.4 UnsupportedDialectFault

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

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

4.5 InvalidExpressionFault

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

[Action]	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault
[Code]	s12:Sender
[Subcode]	wsrt:InvalidExpressionFault
[Reason]	The specified Expression is not valid

[Detail]	`<wsrt:InvalidExpressionSyntax>`
	`  <wsrt:Expression>xs:any</wsrt:Expression> +`
	`</wsrt:InvalidExpressionSyntax>`
	`|`
	`<wsrt:InvalidExpressionValue>`
	`  <wsrt:Expression>xs:any</wsrt:Expression> +`
	`</wsrt:InvalidExpressionValue>`

4.6 GetFault

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

[Action]	http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/fault

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

4.7 ResourceValidityFault

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

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

4.8 FragmentAlreadyExistsFault

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

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

4.9 PutFault

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

The fault detail SHOULD include a wsrt:SideAffects element in the fault detail to indicate whether any changes occurred. A value of “true” indicates some changes occurred; a value of “false” indicates no changes occurred. Absence of the element indicates that changes may have occurred.

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

4.10 PutModeUnsupportedFault

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

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

4.11 CreateFault

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

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

4.12 InvalidMetadataFault

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

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

4.13 MultipartLimitExceededFault

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

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

4.14 InvalidPutSyntaxFault

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

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

5. Security

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

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

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

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

6. Acknowledgements

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

7. References

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

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

[SOAP 1.2]

M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework," June 2003. (See http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.)

[WS-Addressing]
W3C Recommendation, "Web Services Addressing 1.0 (WS-Addressing)," May 2006. (See http://www.w3.org/2005/08/addressing/.)

[WSDL 1.1]
E. Christensen, et al, "Web Services Description Language (WSDL) 1.1," March 2001. (See http://www.w3.org/TR/2001/NOTE-wsdl-20010315.)

[WS-I BP 1.1]
K. Ballinger, et al, "WS-I Basic Profile Version 1.1", April 2006. (See http://www.ws-i.org/Profiles/BasicProfile-1.1.html .)

[WS-MetadataExchange]
K. Ballinger, et al, "Web Services Metadata Exchange (WS-MetadataExchange)", August 2006. (See http://schemas.xmlsoap.org/ws/2004/09/mex.)

[WS-Transfer]
J. Alexander, et al, "Web Services Transfer (WS-Transfer)", March 2006. (See http://www.w3.org/Submission/WS-Transfer/.)

[WS-Security]
OASIS standard, "Web Services Security: SOAP Message Security 1.0" (See http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)

[WS-Trust]
S. Anderson, et al, "Web Services Trust Language (WS-Trust)", February 2005. (See http://schemas.xmlsoap.org/ws/2005/02/trust/)

[WS-SecureConversation]
S. Anderson, et al, "Web Services Secure Conversation Language (WS-SecureConversation)", February 2005. (See http://schemas.xmlsoap.org/ws/2005/02/sc/)

[XML Schema, Part 1]
H. Thompson, et al, "XML Schema Part 1: Structures," October 2004. (See http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/.)

[XML Schema, Part 2]
P. Biron, et al, "XML Schema Part 2: Datatypes," October 2004. (See http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.)

[XPath 1.0]
James Clark, et al, "XML Path Language (XPath) Version 1.0," November 1999, (See http://www.w3.org/TR/xpath)

Appendix I – XPath Level 1

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

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

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

```
(01) <xpath> ::=  <context> <node_sequence>;
(02)
(03) <context> ::= '/' | <>;
```

```
(04)
(05) <node_sequence> ::=
(06)    <element> <optional_collection_operator> <more>;
(07)
(08) <optional_collection_operator> ::= '[' <array_location> ']';
(09) <optional_collection_operator> ::= <>;
(10)
(11) <more> ::= '/' <follower> | <>;
(12)
(13) <follower> ::=
(14)     <attribute> | <text_function> | <node_sequence>;
(15)
(16) <element>        ::= <qualified_name>;
(17) <attribute>      ::= '@' <qualified_name>;
(18)
(19) <qualified_name> ::= <name> <qname_follower>;
(20) <qname_follower> ::= ':' <name> | <>;
(21) <text_function>  ::= "text()" ;
(22) <array_location> ::= NONZERO_DECIMAL_UNSIGNED_INTEGER;
(23) <name>           ::= XML_TOKEN;
```

The terminal tokens which require further lexical specification are NONZERO_DECIMAL_UNSIGNED_INTEGER, whose values are in the subrange (1.. 4294967295), and XML_TOKEN whose values are equivalent to those for the XML Schema type *xs:token*. This grammar is small enough that it can be easily implemented in resource-constrained implementations.

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

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

```
(01) <a>
(02)   <b>
(03)     <c d="30"> 20 </c>
(04)   </b>
(05)   <e>
(06)     <f/>
(07)     <f/>
(08)   </e>
(09) </a>
```

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

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

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

```
(01)  /a
```

The following paths are equivalent:

```
(01)  /a/b
(02)  b
```

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

The <node_sequence> production provides the recursive behavior for the XPath:

```
(01)  /a/b/c
(02)  b/c
```

It also provides for selecting specific repeated elements through the <optional_collection_operator> production:

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

The collection operator only takes unsigned nonzero values, as defined above for NONZERO_DECIMAL_UNSIGNED_INTEGER. Thus, [1] is the first of a repeating series of elements.

The <qualified_name> production allows the XML naming tokens to be either namespace-qualified or unqualified:

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

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

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

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

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

```
(01)  b/c/text()
```

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

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

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

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

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

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

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

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

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

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

Appendix II – Resource Metadata Content

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

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

Table 15: Resource metadata

```
(01)  <wsrt:Metadata>
(02)     <wsrt:Lifetime>lifetime metadata</wsrt:Lifetime> ?
(03)     <wsrt:SupportedDialect>
(04)       dialect metadata
(05)     </wsrt:SupportedDialect> *
(06)   ...
(07)  </wsrt:Metadata>
```

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

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

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

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

II.A Lifecycle metadata

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

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

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

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

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

Table 16: Resource lifecycle metadata

```
(01)  <wsrt:Lifetime>
(02)    ( <wsrt:TerminateAt>
(03)        <wsrt:TerminationTime>xs:dateTime</wsrt:TerminationTime>
(04)        <wsrt:CurrentTime>xs:dateTime</wsrt:CurrentTime>
(05)      </wsrt:TerminateAt> |
(06)      <wsrt:TerminateAfter>xs:duration</wsrt:TerminateAfter> |
(07)      <wsrt:TerminateAfterIdle>
(08)         xs:duration
(09)      </wsrt:TerminateAfterIdle> )
(10)  </wsrt:Lifetime>
```

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

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

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

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

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

II.B Expression Dialect metadata

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

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

Table 17: Resource expression metadata

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

```
(03)            <wsrt:SupportedPutMode>
(04)              ModeType
(05)            </wsrt:SupportedPutMode> *
(06)            <wsrt:MultipartLimit>
(07)              xs:positiveInteger
(08)            </wsrt:MultipartLimit> ?
(09)          </wsrt:SupportedOperation> *
(10)        </wsrt:SupportedDialect>
```

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

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

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

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

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

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

Appendix III – XML Schema

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

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

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

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

```

```
Permission to copy and display the "Web Services Resource Transfer"
Specification, in any medium without fee or royalty is hereby granted,
provided that you include the following on ALL copies of the "Web
Services Resource Transfer" Specification, or portions thereof, that
you make:
 1. A link or URL to the "Web Services Resource Transfer"
    Specification at this location:
       http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer.
 2. The copyright notice as shown in the "Web Services Resource
    Transfer" Specification.

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

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

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

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

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

<xs:schema
      targetNamespace="http://schemas.xmlsoap.org/ws/2006/08/resourceTr
ansfer"
```

```
    xmlns:wsrt="http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:wsa="http://www.w3.org/2005/08/addressing"
     xmlns:wsmex="http://schemas.xmlsoap.org/ws/2004/09/mex"
      elementFormDefault="qualified" blockDefault="#all">

      <xs:import
            namespace="http://www.w3.org/2005/08/addressing"
            schemaLocation=
                "http://www.w3.org/2006/03/addressing/ws-addr.xsd" />
      <xs:import
            namespace="http://schemas.xmlsoap.org/ws/2004/09/mex"
            schemaLocation=
   "http://schemas.xmlsoap.org/ws/2004/09/mex/MetadataExchange.xsd" />


      <!-- ResourceMetadata section -->

      <!-- A wsmex:MetadaSection with a wsmex:Dialect of
           http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer
           contains the following Metadata element
      -->
      <xs:element name="Metadata">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="wsrt:Lifetime" minOccurs="0" />
            <xs:element ref="wsrt:SupportedDialect"
                   minOccurs="0" maxOccurs="unbounded" />
            <xs:any minOccurs="0" maxOccurs="unbounded"
                namespace="##other"  processContents="lax" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>

      <xs:element name="Lifetime">
        <xs:complexType>
          <xs:sequence>
            <xs:choice>
              <xs:element ref="wsrt:TerminateAt"/>
              <xs:element name="TerminateAfter" type="xs:duration" />
              <xs:element name="TerminateAfterIdle"
                 type="xs:duration" />
            </xs:choice>
          </xs:sequence>
        </xs:complexType>
```

```
      </xs:element>

      <xs:element name="TerminateAt">
       <xs:complexType>
         <xs:sequence>
           <xs:element name="TerminationTime" type="xs:dateTime"/>
           <xs:element name="CurrentTime" type="xs:dateTime" />
         </xs:sequence>
       </xs:complexType>
    </xs:element>

      <xs:element name="SupportedDialect">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="wsrt:SupportedOperation"
                                minOccurs="0" maxOccurs="unbounded" />
          </xs:sequence>
          <xs:attribute name='DialectName' type='xs:anyURI'
                                   use='required' />
        </xs:complexType>
      </xs:element>

      <xs:element name="SupportedOperation">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="SupportedPutMode" type="wsrt:ModeType"
                                  minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="MultipartLimit"
               type="xs:positiveInteger" minOccurs="0"/>
          </xs:sequence>
          <xs:attribute name='OperationName' type='xs:anyURI'
                                                    use='required' />
        </xs:complexType>
      </xs:element>

      <!-- Shared Types section -->
      <xs:complexType name='MixedAnyType' final='restriction'
         mixed="true">
        <xs:complexContent mixed="true">
          <xs:restriction base="xs:anyType">
            <xs:sequence>
              <xs:any processContents="lax"
                    minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
          </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
```

```
      <xs:element name="CreateResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="wsrt:ResourceCreated" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>

      <xs:element name="ResourceCreated"
                  type="wsa:EndpointReferenceType" />

      <!-- Put section -->

      <xs:element name='Put'>
        <xs:complexType>
          <xs:sequence>
            <xs:element name='Fragment' type='wsrt:PutFragmentType'
                                maxOccurs='unbounded' />
          </xs:sequence>
          <xs:attribute name='Dialect' type='xs:anyURI' />
        </xs:complexType>
      </xs:element>

      <xs:complexType name='PutFragmentType'>
        <xs:complexContent>
          <xs:extension base="wsrt:FragmentType">
            <xs:attribute name='Mode' type='wsrt:ModeType'
                                use='required' />
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>

      <xs:simpleType name="ModeType" final="restriction">
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="Modify" />
          <xs:enumeration value="Insert" />
          <xs:enumeration value="Remove" />
        </xs:restriction>
      </xs:simpleType>

      <xs:element name="PutResponse">
       <xs:complexType/>
      </xs:element>

      <!-- 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'
```

```
                        maxOccurs='unbounded' />
          </xs:sequence>
        </xs:complexType>
      </xs:element>

      <xs:element name='InvalidExpressionValue'>
        <xs:complexType>
          <xs:sequence>
            <xs:element ref='wsrt:Expression'
                    maxOccurs='unbounded' />
          </xs:sequence>
        </xs:complexType>
      </xs:element>

      <xs:element name='MultipartLimit' type='xs:positiveInteger'/>

      <xs:element name='SideEffects' type='xs:boolean'/>

      <!-- XPath section -->

      <xs:element name='AttributeNode'>
       <xs:complexType>
             <xs:simpleContent>
                  <xs:extension base='xs:string'>
               <xs:attribute name='name' type='xs:QName'/>
             </xs:extension>
              </xs:simpleContent>
       </xs:complexType>
    </xs:element>


    <xs:element name='TextNode' type='xs:string'/>

    <xs:element name='NodeSet' type='wsrt:ResultType'/>

</xs:schema>
```

Appendix IV – WSDL

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

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

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

```
<?xml version="1.0" encoding="utf-8"?>
<!--
Copyright Notice
```

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

Permission to copy and display the "Web Services Resource Transfer"
Specification, in any medium without fee or royalty is hereby granted,
provided that you include the following on ALL copies of the "Web
Services Resource Transfer" Specification, or portions thereof, that
you make:
 1. A link or URL to the "Web Services Resource Transfer"
    Specification at this location:
       http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer.
 2. The copyright notice as shown in the "Web Services Resource
    Transfer" Specification.

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

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

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

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

No other rights are granted by implication, estoppel or otherwise.
-->
```

```
<wsdl:definitions targetNamespace=
   "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer"
   xmlns:wsrt="http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer"
   xmlns:wsa="http://www.w3.org/2005/08/addressing"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wxf="http://schemas.xmlsoap.org/ws/2004/09/transfer"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">

      <wsdl:types>
            <xs:schema>
                  <xs:include schemaLocation=
   "http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/wsrt.xsd" />
            </xs:schema>
      </wsdl:types>

      <wsdl:message name="CreateRequestMessage">
            <wsdl:part name="Body" element="wsrt:Create" />
      </wsdl:message>

      <wsdl:message name="CreateResponseMessage">
            <wsdl:part name="Body" element="wsrt:CreateResponse" />
      </wsdl:message>

      <wsdl:message name="GetRequestMessage">
            <wsdl:part name="Body" element="wsrt:Get" />
      </wsdl:message>

      <wsdl:message name="GetResponseMessage">
            <wsdl:part name="Body" element="wsrt:GetResponse" />
      </wsdl:message>

      <wsdl:message name="PutRequestMessage">
            <wsdl:part name="Body" element="wsrt:Put" />
      </wsdl:message>

      <wsdl:message name="PutResponseMessage">
            <wsdl:part name="Body" element="wsrt:PutResponse" />
      </wsdl:message>

      <wsdl:portType name="ResourceInterface">
        <wsdl:documentation>
          This port type contains the Get and Put
          operations defined in WS-ResourceTransfer.
        </wsdl:documentation>
```

```
        <wsdl:operation name="Get">
          <wsdl:input message="wsrt:GetRequestMessage"
            wsa:Action=
               "http://schemas.xmlsoap.org/ws/2004/09/transfer/Get" />
          <wsdl:output message="wsrt:GetResponseMessage"
            wsa:Action=
       "http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse" />
        </wsdl:operation>
        <wsdl:operation name="Put">
          <wsdl:input message="wsrt:PutRequestMessage"
            wsa:Action=
               "http://schemas.xmlsoap.org/ws/2004/09/transfer/Put" />
          <wsdl:output message="wsrt:PutResponseMessage"
            wsa:Action=
       "http://schemas.xmlsoap.org/ws/2004/09/transfer/PutResponse" />
            </wsdl:operation>
      </wsdl:portType>

      <wsdl:portType name="ResourceFactoryInterface">
        <wsdl:documentation>
          This port type contains the Create operation
          defined in WS-ResourceTransfer.
        </wsdl:documentation>
        <wsdl:operation name="Create">
          <wsdl:input message="wsrt:CreateRequestMessage"
             wsa:Action=
            "http://schemas.xmlsoap.org/ws/2004/09/transfer/Create" />
          <wsdl:output message="wsrt:CreateResponseMessage"
             wsa:Action=
    "http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse" />
        </wsdl:operation>
      </wsdl:portType>

</wsdl:definitions>
```