Web Services Transfer (WS-Transfer)

27 September 2006

Authors:
Jan Alexander, Systinet
Don Box, Microsoft
Luis Felipe Cabrera, Microsoft
Dave Chappell, Sonic Software
Glen Daniels, Sonic Software
Radovan Janecek, Systinet
Chris Kaler, Microsoft
Brad Lovering, Microsoft
Raymond McCollum, Microsoft
David Orchard, BEA
Savas Parastatidis, Microsoft
Jeffrey Schlimmer, Microsoft
Igor Sedukhin, Computer Associates
John Shewchuk, Microsoft

Copyright Notice

Abstract

This specification describes a general SOAP-based protocol for accessing XML representations of Web service-based resources.

Status of this document

WS-Transfer and related specifications are provided as-is and for review and evaluation only. BEA Systems, Computer Associates, Microsoft, Sonic Software, and Systinet make no warrantees or representations regarding the specifications in any manner whatsoever.

Table of Contents

1. Introduction

This specification defines a mechanism for acquiring XML-based representations of entities using the Web service infrastructure. It defines two types of entities:

- Resources, which are entities addressable by an endpoint reference that provide an XML representation
- Resource factories, which are Web services that can create a new resource from an XML representation

Specifically, it defines two operations for sending and receiving the representation of a given resource and two operations for creating and deleting a resource and its corresponding representation.

It should be noted that the state maintenance of a resource is at most subject to the "best efforts" of the hosting server. When a client receives the server's acceptance of a request to create or update a resource, it can reasonably expect that the resource now exists at the confirmed location and with the confirmed representation, but this is not a guarantee, even in the absence of any third parties. The server may change the representation of a resource, may remove a resource entirely, or may bring back a resource that was deleted.

For instance, the server may store resource state information on a disk drive. If that drive crashes and the server recovers state information from a backup tape, changes that occurred after the backup was made will be lost.

A server may have other operational processes that change resource state information. A server may run a background process that examines resources for objectionable content and deletes any such resources it finds. A server may purge resources that have not been accessed for some period of time. A server may apply storage quotas that cause it to occasionally purge resources.

In essence, the confirmation by a service of having processed a request to create, modify, or delete a resource implies a commitment only at the instant that the confirmation was generated. While the usual case should be that resources are long-lived and stable, there are no guarantees, and clients should code defensively.

There is no requirement for uniformity in resource representations between the messages defined in this specification. For example, the representations required by Create or Put may differ from the representation returned by Get, depending on the semantic requirements of the service. Additionally, there is no requirement that the resource content is fixed for any given endpoint reference. The resource content may vary based on environmental factors, such as the security context, time of day, configuration, or the dynamic state of the service.

As per the SOAP processing model, other specifications may define SOAP headers which may be optionally added to request messages to require the transfer of subsets or the application of transformations of the resource associated with the endpoint reference. When the Action URIs defined by this specification are used, such extension specifications must also allow the basic processing models defined herein.

1.1 Requirements

This specification intends to meet the following requirements:

- Provide a SOAP-based protocol for managing resources and their representations.
- Minimize additional mechanism beyond the current Web Services architecture.

2. Notations and Terminology

This section specifies the notations, namespaces, and terminology used in this specification.

2.1 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 normative outlines for messages:

- The syntax appears as an XML instance, but values in italics indicate data types instead of 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.
- An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content. 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. If a receiver does not recognize an extension, the receiver SHOULD ignore it.
- XML namespace prefixes (see table 1) are used to indicate the namespace of the element being defined.

2.2 XML Namespaces

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

```
http://schemas.xmlsoap.org/ws/2004/09/transfer
```

Table 1 lists XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

Table 1: Prefixes and XML namespaces used in this specification

Prefix	XML Namespace	Specification(s)
wxf	http://schemas.xmlsoap.org/ws/2004/09/transfer	This specification
s11	http://schemas.xmlsoap.org/soap/envelope/	SOAP 1.1 [SOAP 1.1]
s12	http://www.w3.org/2003/05/soap-envelope	SOAP 1.2 [SOAP 1.2]
s	Either of s11 or s12	SOAP
wsa	Either of wsa1 or wsa2	WS-Addressing
xs	http://www.w3.org/2001/XMLSchema	XML Schema [Part 1, 2]
wsdl	http://schemas.xmlsoap.org/wsdl	WSDL/1.1 [WSDL 1.1]
wsa1	http://schemas.xmlsoap.org/ws/2004/08/addressing	WS-Addressing Submission [WS-Addressing Submission]
wsa2	http://www.w3.org/2005/08/addressing/	WS-Addressing 1.0 Core [WS-Addressing 1.0] and [WS-Addressing 1.0 SOAP]

2.3 Terminology

Resource

> A Web service that is addressable by an endpoint reference as defined in WS-Addressing and that can be represented by an XML Infoset using the Get and Put operations defined in this specification

Resource factory

> A Web service that is capable of creating new resources using the Create operation defined in this specification

2.4 Compliance

An implementation is not compliant with this specification if it fails to satisfy one or more of the MUST or REQUIRED level requirements defined herein. A SOAP Node MUST NOT use the XML namespace identifier for this specification (listed in Section 2.2) within SOAP envelopes unless it is compliant with this specification.

Specifically, a compliant SOAP Node that implements a resource MUST provide the Get operation as defined in this specification, and MAY provide the Put and Delete operations.

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

Conformant implementations may support either or both of [WS-Addressing Submission] or [WS-Addressing 1.0]. In any given request-response message exchange, the responses generated by the service server MUST use the same WS-Addressing namespace binding that was used in the request.

3. Resource Operations

3.1 Get

This specification defines one Web service operation (Get) for fetching a one-time snapshot of the representation of a resource.

The Get request message MUST be of the following form:

```
<s:Envelope …>
  <s:Header …>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body ...>
     ...
  </s:Body>
</s:Envelope>
```

The following describes additional, normative constraints on the outline listed above:

`/s:Envelope/s:Header/wsa:Action`

> This required element MUST contain the value http://schemas.xmlsoap.org/ws/2004/09/transfer/Get. If a SOAP Action URI is also present in the underlying transport, its value MUST convey the same value.

A Get request MUST be targeted at the resource whose representation is desired as described in Section 2 of this specification.

There are no body blocks defined by default for a Get Request. As per the SOAP processing model, other specifications may introduce various types of extensions to the semantics of this message which are enabled through headers tagged with s*:mustUnderstand="true"*. Such extensions may define how resource or subsets of it are to be retrieved or transformed prior to retrieval.   Specifications which define such extensions MUST allow processing the basic Get request message without those extensions.   Since the response may not be sent to the original sender, extension specifications should consider adding a corresponding SOAP header value in the response to signal to the receiver that the extension is being used.

Implementations may respond with a fault message using the standard fault codes defined in WS-Addressing (e.g., wsa:ActionNotSupported). Other components of the outline above are not further constrained by this specification.

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

```
<s:Envelope …>
  <s:Header …>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body …>
    xs:any
     ...
  </s:Body>
</s:Envelope>
```

The following describes additional, normative constraints on the outline listed above:

/s:Envelope/s:Header/wsa:Action

> This required element MUST contain the value http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse. If a SOAP Action URI is also present in the underlying transport, its value MUST convey the same value.

/s:Envelope/s:Body/child::*[position()=1]

> The representation itself MUST be the initial child element of the SOAP:Body element of the response message.   The presence of subsequent child elements is service-specific and MAY be controlled by the presence or extension-specific SOAP headers in the original request.

Other components of the outline above are not further constrained by this specification.

The following shows a sample SOAP envelope containing a Get request:

```
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:xxx="http://fabrikam123.example.com/resource-model" >
  <s:Header>
```

```
    <wsa:ReplyTo>
      <wsa:Address>
        soap://www.fabrikam123.example.org/pullport
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>soap://www.example.org/repository</wsa:To>
    <xxx:CustomerID>732199</xxx:CustomerID>
    <xxx:Region>EMEA</xxx:Region>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
    </wsa:Action>
    <wsa:MessageID>
      uuid:00000000-0000-0000-C000-000000000046
    </wsa:MessageID>
  </s:Header>
  <s:Body/>
</s:Envelope>
```

The following shows the corresponding response message:

```
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:xxx="http://fabrikam123.example.com/resource-model" >
  <s:Header>
    <wsa:To>soap://www.fabrikam123.example.org/pullport</wsa:Address>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
    </wsa:Action>
    <wsa:MessageID>
      uuid:0000010e-0000-0000-C000-000000000046
    </wsa:MessageID>
    <wsa:RelatesTo>
      uuid:00000000-0000-0000-C000-000000000046
    </wsa:RelatesTo>
  </s:Header>
  <s:Body>
    <xxx:Customer>
      <xxx:first>Roy</xxx:first><xxx:last>Hill</xxx:last>
      <xxx:address>123 Main Street</xxx:address>
      <xxx:city>Manhattan Beach</xxx:city>
      <xxx:state>CA</xxx:state>
      <xxx:zip>90266</xxx:zip>
    </xxx:Customer>
  </s:Body>
</s:Envelope>
```

In this example, the representation of the resource is the following XML element:

```
  <xxx:Customer>
    <xxx:first>Roy</xxx:first><xxx:last>Hill</xxx:last>
    <xxx:address>123 Main Street</xxx:address>
    <xxx:city>Manhattan Beach</xxx:city>
    <xxx:state>CA</xxx:state>
    <xxx:zip>90266</xxx:zip>
  </xxx:Customer>
```

3.2 Put

This specification defines one Web service operation (Put) for updating a resource by providing a replacement representation. A resource MAY accept updates that provide different XML representations than that returned by the resource; in such a case, the semantics of the update operation is defined by the resource.

The Put request message MUST be of the following form:

```
<s:Envelope …>
  <s:Header …>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Put
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body…>
    xs:any
    ...
  </s:Body>
</s:Envelope>
```

The following describes additional, normative constraints on the outline listed above:

`/s:Envelope/s:Header/wsa:Action`

> This required element MUST contain the value http://schemas.xmlsoap.org/ws/2004/09/transfer/Put. If a SOAP Action URI is also present in the underlying transport, its value MUST convey the same value.

`/s:Envelope/s:Body/child::*[position()=1]`

> The representation to be used for the update MUST be the initial child element of the s:Body element of the request message. All other children SHOULD be ignored by the service.

A Put request MUST be targeted at the resource whose representation is desired to be replaced, as described in Section 2 of this specification. As per the SOAP processing model, other specifications MAY introduce various types of extensions to this message which are enabled through headers tagged with s*:mustUnderstand="true"*. Such extensions may require that a full or partial update should be accomplished using symbolic, instruction-based, or other methodologies.

Extension specifications MAY also define extensions to the original Put request, enabled by OPTIONAL SOAP headers, which control the nature of the response, as discussed in remarks on the PutResponse below.

Specifications which define any of these extensions MUST allow processing the Put message without such extensions.

In addition to the standard fault codes defined in WS-Addressing, implementations MAY use the fault code wxf:InvalidRepresentation if the presented representation is invalid for the target resource. See Section 5. Other components of the outline above are not further constrained by this specification.

A successful Put operation updates the current representation associated with the targeted resource.

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

```
<s:Envelope …>
  <s:Header …>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/PutResponse
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body …>
    xs:any ?
  </s:Body>
</s:Envelope>
```

`/s:Envelope/s:Header/wsa:Action`

> This required element MUST contain the value http://schemas.xmlsoap.org/ws/2004/09/transfer/PutResponse. If a SOAP Action URI is also present in the underlying transport, its value MUST convey the same value.

`/s:Envelope/s:Body/child::*[position()=1]`

> By default, a service MUST return the current representation of the resource as the initial child of the s:Body element if the updated representation differs from the representation sent in the Put request message. The presence of additional child elements which contain other information pertaining to the update is service-specific.
>
> As an optimization and as a service to the requester, the s:Body element of the response message SHOULD be empty if the updated representation does not differ from the representation sent in the Put request message; that is, if the service accepted the new representation verbatim.

Such a response (an empty s:Body) implies that the update request was successful in its entirety (assuming no intervening mutating operations are performed). A service MAY return the current representation of the resource as the initial child of the s:Body element even in this case, however.

Extension specifications MAY define extensions to the original Put request, enabled by OPTIONAL header values, which specifically control the presence, absence, or format of the updated representation or other child elements in the PutResponse in order to optimize the response. In the absence of such headers, the behavior MUST be as described above. Specifications which define any of these extensions MUST allow processing the Put message without such extensions. Since the response may not be sent to the original sender, extension specifications should consider adding a corresponding SOAP header value in the response to signal to the receiver that the extension is being used.

Other components of the outline above are not further constrained by this specification.

The following shows a sample SOAP envelope containing a Put request:

```
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:xxx="http://fabrikam123.example.com/resource-model" >
  <s:Header>
    <wsa:ReplyTo>
      <wsa:Address>
        soap://www.fabrikam123.example.org/sender
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>soap://www.example.org/pushport</wsa:To>
    <xxx:CustomerID>732199</xxx:CustomerID>
    <xxx:Region>EMEA</xxx:Region>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Put
    </wsa:Action>
    <wsa:MessageID>
      uuid:00000000-0000-0000-C000-000000000047
    </wsa:MessageID>
  </s:Header>
  <s:Body>
    <xxx:Customer>
      <xxx:first>Roy</xxx:first><xxx:last>Hill</xxx:last>
      <xxx:address>321 Main Street</xxx:address>
      <xxx:city>Manhattan Beach</xxx:city>
      <xxx:state>CA</xxx:state>
      <xxx:zip>90266</xxx:zip>
    </xxx:Customer>
  </s:Body>
</s:Envelope>
```

The following shows the corresponding response message indicating success:

```
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:xxx="http://fabrikam123.example.com/resource-model" >
  <s:Header>
    <wsa:To>soap://www.fabrikam123.example.org/sender</wsa:Address>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/PutResponse
    </wsa:Action>
    <wsa:MessageID>
      uuid:0000010e-0000-0000-C000-000000000047
    </wsa:MessageID>
    <wsa:RelatesTo>
      uuid:00000000-0000-0000-C000-000000000047
    </wsa:RelatesTo>
  </s:Header>
  <s:Body/>
</s:Envelope>
```

3.3 Delete

This specification defines one Web service operation (Delete) for deleting a resource in its entirety.

Extension specifications MAY define extensions to the Delete request, enabled by OPTIONAL header values, which specifically control preconditions for the Delete to succeed and which may control the nature or format of the response. Since the response may not be sent to the original sender, extension specifications should consider adding a corresponding SOAP header value in the response to signal to the receiver that the extension is being used.

The Delete request message MUST be of the following form:

```
<s:Envelope …>
  <s:Header …>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Delete
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body … />
</s:Envelope>
```

The following describes additional, normative constraints on the outline listed above:

`/s:Envelope/s:Header/wsa:Action`

This required element MUST contain the value http://schemas.xmlsoap.org/ws/2004/09/transfer/Delete. If a SOAP Action URI is also present in the underlying transport, its value MUST convey the same value.

A Delete request MUST be targeted at the resource to be deleted as described in Section 2 of this specification.

There are no body blocks defined for a Delete Request.

Implementations may respond with a fault message using the standard fault codes defined in WS-Addressing (e.g., `wsa:ActionNotSupported`). Other components of the outline above are not further constrained by this specification.

A successful Delete operation invalidates the current representation associated with the targeted resource.

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

```
<s:Envelope …>
  <s:Header …>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/DeleteResponse
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body …>
    ...
  </s:Body>
</s:Envelope>
```

`/s:Envelope/s:Header/wsa:Action`

This required element MUST contain the value http://schemas.xmlsoap.org/ws/2004/09/transfer/DeleteResponse. If a SOAP Action URI is also present in the underlying transport, its value MUST convey the same value.

By default, there are no s:Body blocks defined for a Delete response.   Specifications which define extensions for use in the original Delete request which control the format of the response MUST allow processing the Delete message without such extensions.

Other components of the outline above are not further constrained by this specification.

The following shows a sample SOAP envelope containing a Delete request:

```
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:xxx="http://fabrikam123.example.com/resource-model" >
  <s:Header>
```

```
    <wsa:ReplyTo>
      <wsa:Address>
        soap://www.fabrikam123.example.org/sender
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>soap://www.example.org/pushport</wsa:To>
    <xxx:CustomerID>732199</xxx:CustomerID>
    <xxx:Region>EMEA</xxx:Region>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Delete
    </wsa:Action>
    <wsa:MessageID>
      uuid:00000000-0000-0000-C000-000000000049
    </wsa:MessageID>
  </s:Header>
  <s:Body/>
</s:Envelope>
```

The following shows the corresponding response message indicating success:

```
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:xxx="http://fabrikam123.example.com/resource-model" >
  <s:Header>
    <wsa:To>soap://www.fabrikam123.example.org/sender</wsa:Address>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/DeleteResponse
    </wsa:Action>
    <wsa:MessageID>
      uuid:0000010e-0000-0000-C000-000000000049
    </wsa:MessageID>
    <wsa:RelatesTo>
      uuid:00000000-0000-0000-C000-000000000049
    </wsa:RelatesTo>
  </s:Header>
  <s:Body/>
</s:Envelope>
```

4. Resource Factory Operations

4.1 Create

This specification defines one Web service operation (Create) for creating a resource and providing its initial representation. In some cases, the initial representation MAY constitute the representation of a logical constructor for the resource and may thus differ structurally from the representation returned by Get or the one required by Put. This is because the parameterization requirement for creating a resource is often distinct from the steady-state representation of the resource. Implementations should provide metadata which describes the use of the representation and how it relates to the resource which is created, but such mechanisms are beyond the scope of this specification. 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 service-determined endpoint reference that is returned in the response message.

The Create request message MUST be of the following form:

```
<s:Envelope …>
  <s:Header …>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body …>
    xs:any
    ...
  </s:Body>
</s:Envelope>
```

The following describes additional, normative constraints on the outline listed above:

`/s:Envelope/s:Header/wsa:Action`

> This required element MUST contain the value http://schemas.xmlsoap.org/ws/2004/09/transfer/Create. If a SOAP Action URI is also present in the underlying transport, its value MUST convey the same value.

`/s:Envelope/s:Body/child::*[position()=1]`

> The first child element of the s:Body element MUST NOT be omitted. The contents of this element are service-specific, and MAY contain the literal initial resource representation, a representation of the constructor for the resource, or other instructions for creating the resource. All other children SHOULD be ignored by the service.

Extensions specifications MAY also define extensions to the original Create request, enabled by OPTIONAL SOAP headers, which constrain the nature of the response, as discussed in remarks on the CreateResponse below. Similarly, they may require headers which control the interpretation of the s:Body as part of the resource creation process.

Such specifications MUST also allow processing the Create message without such extensions.

A Create request MUST be targeted at a resource factory capable of creating the desired new resource. This factory is distinct from the resource being created (which by definition does not exist prior to the successful processing of the Create request message).

In addition to the standard fault codes defined in WS-Addressing, implementations MAY use the fault code wfx:InvalidRepresentation if the presented representation is invalid for the target resource. See Section 5.

Other components of the outline above are not further constrained by this specification.

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

```
<s:Envelope …>
  <s:Header …>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body …>
    <wxf:ResourceCreated>endpoint-reference</wxf:ResourceCreated>
    xs:any ?
  </s:Body>
</s:Envelope>
```

`/s:Envelope/s:Header/wsa:Action`

This required element MUST contain the value http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse. If a SOAP Action URI is also present in the underlying transport, its value MUST convey the same value.

`/s:Envelope/s:Body/wxf:ResourceCreated`

This required element MUST contain a resource reference for the newly created resource. This resource reference, represented as an endpoint reference as defined in WS-Addressing, MUST identify the resource for future Get, Put, and Delete operations.

`/s:Envelope/s:Body/child::*[position()=2]`

By default, a service MUST return the current representation of the new resource as the second child of the s:Body element if the created representation logically differs from the representation sent in the Create request message.   That is, the initial representation is returned if one or more values present in Create message was specifically overridden with a different value during resource creation.   If default values are used to complete a resource creation which were not present in the Create message, then this does not constitute a logical difference. The presence of additional child elements which contain other information pertaining to the result of the Create operation is service-specific.

As an optimization and as a service to the requestor, the s:Body element of the response message SHOULD be empty, other than the ResourceCreated element, if the created representation does not logically differ from the representation sent in the Create request message; that is, if the service accepted the new representation or creation instructions verbatim. Such a response indicates that the request was completely successful (assuming no intervening mutating operations are performed). A service MAY return the current representation of the resource as the initial child of the s:Body element even in this case, however.

Extension specifications MAY define extensions to the original Create request, enabled by OPTIONAL header values, which specifically control the presence, absence, or format of the initial representation or other child elements in the CreateResponse. These headers MAY override the default behavior if they are marked with *s:mustUnderstand="true"*. In the absence of such OPTIONAL headers, the behavior MUST be as described in the previous paragraphs. Since the response may not be sent to the original sender, extension specifications should consider adding a corresponding SOAP header value in the response to signal to the receiver that the extension is being used.

Other components of the outline above are not further constrained by this specification.

The following shows a sample SOAP envelope containing a Create request:

```
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:xxx="http://fabrikam123.example.com/resource-model" >
  <s:Header>
    <wsa:ReplyTo>
      <wsa:Address>
        soap://www.fabrikam123.example.org/sender
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>soap://www.example.org/pushport/CustomerSpace</wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
    </wsa:Action>
    <wsa:MessageID>
      uuid:00000000-0000-0000-C000-000000000048
    </wsa:MessageID>
  </s:Header>
  <s:Body>
    <xxx:Customer>
      <xxx:first>Roy</xxx:first><xxx:last>Hill</xxx:last>
      <xxx:address>123 Main Street</xxx:address>
      <xxx:city>Manhattan Beach</xxx:city>
      <xxx:state>CA</xxx:state>
      <xxx:zip>90266</xxx:zip>
    </xxx:Customer>
  </s:Body>
</s:Envelope>
```

The following shows the corresponding response message indicating success:

```
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wxf="http://schemas.xmlsoap.org/ws/2004/09/transfer"
    xmlns:xxx="http://fabrikam123.example.com/resource-model" >
  <s:Header>
    <wsa:To>soap://www.fabrikam123.example.org/sender</wsa:Address>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse
    </wsa:Action>
    <wsa:MessageID>
      uuid:0000010e-0000-0000-C000-000000000048
    </wsa:MessageID>
    <wsa:RelatesTo>
      uuid:00000000-0000-0000-C000-000000000048
    </wsa:RelatesTo>
  </s:Header>
  <s:Body>
    <wxf:ResourceCreated>
      <wsa:Address>soap://www.example.org/pushport</wsa:Address>
      <wsa:ReferenceParameters>
        <xxx:CustomerID>732199</xxx:CustomerID>
        <xxx:Region>EMEA</xxx:Region>
      </wsa:ReferenceParameters>
    </wxf:ResourceCreated>
  </s:Body>
</s:Envelope>
```

5. Faults

All fault messages defined in this specification MUST be sent according to the rules and usage described in [WS-Addressing 1.0 - SOAP Binding] Section 6 for encoding SOAP 1.1 and SOAP 1.2 faults. The [action] property below SHOULD be used for faults defined in this specification:

http://schemas.xmlsoap.org/ws/2004/09/transfer/fault

5.1 InvalidRepresentation

This fault is returned when an incorrect representation is sent in a wxf:Put or wxf:Create message.

[Code]	s:Sender
[Subcode]	wxf:InvalidRepresentation
[Reason]	The supplied representation is invalid
[Detail]	none

6. Security Considerations

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, WS-Security timestamp, and any header blocks resulting from a `<wsa:ReferenceParameters>` in references 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 is 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.

It is recommended that the security considerations of WS-Security also be considered.

While a comprehensive listing of attacks is not feasible, the following list summarizes common classes of attacks that apply to this protocol and identifies the mechanism(s) to prevent/mitigate the attacks.

- **Replay** — Messages, or portions of messages, can be replayed in an attempt to gain access or disrupt services. Freshness checks such as timestamps, digests, and sequences can be used to detect duplicate messages.
- **Invalid tokens** — There are a number of token attacks including certificate authorities, false signatures, and PKI attacks. Care should be taken to ensure each token is valid (usage window, digest, signing authority, revocation, …), and that the appropriate delegation policies are in compliance.
- **Man-in-the-middle** — The message exchanges in this specification could be subject to man-in-the-middle attacks so care should be taken to reduce possibilities here such as establishing a secure channel and verifying that the security tokens user represent identities authorized to speak for, or on behalf of, the desired resource reference.
- **Message alteration** — Alteration is prevented by including signatures of the message information using WS-Security. Care should be taken to review message part references to ensure they haven't been forged (e.g. ID duplication).
- **Message disclosure** — Confidentiality is preserved by encrypting sensitive data using WS-Security.
- **Key integrity** — Key integrity is maintained by using the strongest algorithms possible (by comparing secured policies — see [WS-Policy] and [WS-SecurityPolicy]) and by using derived keys ([WS-SecureConversation]).

- **Authentication** — Authentication is established using the mechanisms described in WS-Security and WS-Trust. Each message is authenticated using the mechanisms described in WS-Security.
- **Accountability** — Accountability is a function of the type of and string of the key and algorithms being used. In many cases, a strong symmetric key provides sufficient accountability. However, in some environments, strong PKI signatures are required.
- **Availability** — All reliable messaging services are subject to a variety of availability attacks. Replay detection is a common attack and it is recommended that this be addressed by the mechanisms described in WS-Security. Other attacks, such as network-level denial of service attacks are harder to avoid and are outside the scope of this specification. That said, care should be taken to ensure that minimal state is saved prior to any authenticating sequences.

7. Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams, including:

- Erik Christensen, Microsoft
- Doug Davis, IBM
- Henrik Frystyk Nielsen, Microsoft
- Omri Gazitt, Microsoft
- Kirill Gavrylyuk, Microsoft
- Alan Geller
- Martin Gudgin, Microsoft
- Andrew Layman, Microsoft
- Steve Millet, Microsoft
- Bryan Murray, Hewlett-Packard
- Brian Reistad, Microsoft
- Ian Robinson, IBM
- Vijay Tewari, Intel
- Marvin Theimer, Microsoft
- William Vambenepe, Hewlett-Packard

8. References

[RFC 2119]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, 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 Submission]
D. Box, et al, "Web Services Addressing (WS-Addressing)," August 2004. (See http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/.)
[WS-Addressing 1.0 - Core]
M. Gudgin, et al, "Web Services Addressing 1.0 - Core," May 2006. (See http://www.w3.org/TR/2006/REC-ws-addr-core-20060509)
[WS-Addressing 1.0 - SOAP Binding]
M. Gudgin, et al, "Web Services Addressing 1.0 - SOAP Binding," May 2006. (See http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/)
[WS-Policy]
S. Bajaj, et al, "Web Services Policy Framework (WS-Policy)," September 2004. (See http://schemas.xmlsoap.org/ws/2004/09/policy.)
[WS-SecureConversation]
S. Anderson, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," February 2005. (See http://schemas.xmlsoap.org/ws/2005/02/sc.)
[WS-Security]
A. Nadalin, et al, "Web Services Security: SOAP Message Security 1.0," March 2004. (See http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.)
[WS-SecurityPolicy]
G. Della-Libera, et al, "Web Services Security Policy Language (WS-SecurityPolicy), Version 1.1," July 2005. (See http://schemas.xmlsoap.org/ws/2005/07/securitypolicy.)
[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.)
[XML Infoset]
J. Cowan, et al, "XML Information Set," February 2004. (See http://www.w3.org/TR/2004/REC-xml-infoset-20040204/.)
[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/.)

Appendix I — XSD

A normative copy of the XML Schema [XML Schema Part 1, Part 2] for this specification may be retrieved by resolving the XML namespace URI for this specification (listed in Section 2.2 XML Namespaces).

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

```
<xs:schema
  targetNamespace="http://schemas.xmlsoap.org/ws/2004/09/transfer"
  xmlns:tns="http://schemas.xmlsoap.org/ws/2004/09/transfer"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wsa04="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wsa10="http://www.w3.org/2005/08/addressing"
  elementFormDefault="qualified"
  blockDefault="#all" >

  <xs:import
    namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"

schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing/addressing.x
sd" />

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

  <xs:complexType name="AnyXmlType" >
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="AnyXmlOptionalType" >
    <xs:sequence>
      <xs:any minOccurs="0" namespace="##other" processContents="lax" />
    </xs:sequence>
  </xs:complexType>

<!--
The type of the ResourceCreated is effectively
the union of wsa04:EndpointReferenceType and
wsa10:EndpointReferenceType. Unfortunately, xs:union only
works for simple types. As a result, we have to define
the element in an unvalidated way to accommodate either
addressing type.
-->

  <xs:element name="ResourceCreated">
    <xs:complexType>
      <xs:sequence>
        <xs:any minOccurs='1' maxOccurs='unbounded' processContents='skip'
namespace='##other' />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="CreateResponseType" >
    <xs:sequence>
      <xs:element ref="tns:ResourceCreated" />
      <xs:any minOccurs="0" namespace="##other" processContents="lax" />
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Appendix II — WSDL

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

```
http://schemas.xmlsoap.org/ws/2004/09/transfer/transfer.wsdl
```

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

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

  <wsdl:types>
    <xs:schema>
      <xs:import
        namespace="http://schemas.xmlsoap.org/ws/2004/09/transfer"

schemaLocation="http://schemas.xmlsoap.org/ws/2004/09/transfer/transfer.xsd"
        />
    </xs:schema>
  </wsdl:types>

  <wsdl:message name="EmptyMessage"/>
  <wsdl:message name="AnyXmlMessage">
    <wsdl:part name="Body" type="tns:AnyXmlType"/>
  </wsdl:message>
  <wsdl:message name="OptionalXmlMessage">
    <wsdl:part name="Body" type="tns:AnyXmlOptionalType"/>
  </wsdl:message>
  <wsdl:message name="CreateResponseMessage">
    <wsdl:part name="Body" type="tns:CreateResponseType"/>
  </wsdl:message>

  <wsdl:portType name="Resource">
    <wsdl:documentation>
      This port type defines a resource that may be read,
      written, and deleted.
    </wsdl:documentation>
    <wsdl:operation name="Get">
      <wsdl:input
        message="tns:OptionalXmlMessage"
        wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <wsdl:output
        message="tns:AnyXmlMessage"

wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse" />
    </wsdl:operation>
    <wsdl:operation name="Put">
      <wsdl:input
        message="tns:AnyXmlMessage"
        wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/Put" />
      <wsdl:output
        message="tns:OptionalXmlMessage"
```

```
wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/PutResponse" />
    </wsdl:operation>
    <wsdl:operation name="Delete">
      <wsdl:input
        message="tns:EmptyMessage"
        wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/Delete" />
      <wsdl:output
        message="tns:OptionalXmlMessage"

wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/DeleteResponse" />
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:portType name="ResourceFactory">
    <wsdl:documentation>
      This port type defines a Web service that can create new
      resources.
    </wsdl:documentation>
    <wsdl:operation name="Create">
      <wsdl:input
        message="tns:AnyXmlMessage"
        wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/Create" />
      <wsdl:output
        message="tns:CreateResponseMessage"

wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse" />
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```