Web Services Metadata Exchange (WS-MetadataExchange)

March 2004

Authors

Keith Ballinger, Microsoft
Don Box, Microsoft
Francisco Curbera (Editor), IBM
Steve Graham, IBM
Canyang Kevin Liu, SAP
Brad Lovering, Microsoft
Anthony Nadalin, IBM
Mark Nottingham, BEA Systems
David Orchard, BEA Systems
Claus von Riegen, SAP
Jeffrey Schlimmer (Editor), Microsoft
John Shewchuk, Microsoft
Greg Truty, IBM
Sanjiva Weerawarana, IBM

Copyright Notice

(c) 2004 BEA Systems Inc., International Business Machines Corporation, Microsoft Corporation, Inc, and SAP AG. All rights reserved.

BEA Systems, IBM, Microsoft, and SAP (collectively, the "Authors") hereby grant you permission to copy and display the WS-MetadataExchange Specification (the Specification), in any medium without fee or royalty, provided that you include the following on ALL copies of the Specification, or portions thereof, that you make:

1. A link or URL to the Specification at this location.

2. The copyright notice as shown in the WS-MetadataExchange Specification.

EXCEPT FOR THE COPYRIGHT LICENSE GRANTED ABOVE, THE AUTHORS DO NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY, INCLUDING PATENTS, THEY OWN OR CONTROL.

THE 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 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 SPECIFICATION.

The Specification may change before final release and you are cautioned against relying on the content of this Specification.

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

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

Abstract

This specification defines messages to retrieve specific types of metadata associated with an endpoint.

Composable Architecture

By using the XML, SOAP [SOAP 1.1, SOAP 1.2], and WSDL [WSDL 1.1] extensibility models, the Web services specifications (WS-*) are designed to be composed with each other to provide a rich set of tools to provide security in the Web services environment. This specification specifically relies on other Web services specifications to provide secure, reliable, and/or transacted message delivery and to express Web service and client policy.

Status

This specification is an initial public draft release and is provided for review and evaluation only. The authors hope to solicit your contributions and suggestions in the near future. The authors make no warrantees or representations regarding the specifications in any manner whatsoever.

Table of Contents

1. Introduction

Web services use metadata to describe what other endpoints need to know to interact with them. Specifically, WS-Policy [WS-Policy] describes the capabilities, requirements, and general characteristics of Web services; WSDL [WSDL 1.1] describes abstract message operations, concrete network protocols, and endpoint addresses used by Web services; XML Schema [XML Schema Part 1, Part 2] describes the structure and contents of XML-based messages received and sent by Web services.

To bootstrap communication with a Web service, this specification defines three request-response message pairs to retrieve these three types of metadata: one retrieves the WS-Policy associated with the receiving endpoint or with a given target namespace, another retrieves either the WSDL associated with the receiving endpoint or with a given target namespace, and a third retrieves the XML Schema with a given target namespace. Together these messages allow efficient, incremental retrieval of a Web service's metadata.

1.1 Requirements

This specification intends to meet the following requirements:

- Define a bootstrap mechanism for metadata-driven [XML Schema, WSDL, and WS-Policy] message exchange.
- Leverage other Web service specifications for secure, reliable, transacted message delivery.
- Support both SOAP 1.1 [SOAP 1.1] and SOAP 1.2 [SOAP 1.2] Envelopes.
- Enable description in WSDL 1.1 [WSDL 1.1].

1.2 Example

Table 1 illustrates a sample Get Policy request.

Table 1: Sample Get Policy request message.

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/03/addressing'
(04)     xmlns:wsx='http://schemas.xmlsoap.org/ws/2004/03/mex' >
(05)   <s12:Header>
(06)     <wsa:Action>
(07)         http://schemas.xmlsoap.org/ws/2004/03/mex/GetPolicy/Request
(08)     </wsa:Action>
(09)     <wsa:MessageID>
(10)         uuid:73d7edfc-5c3c-49b9-ba46-2480caee43e9
```

```
(11)      </wsa:MessageID>
(12)      <wsa:ReplyTo>
(13)        <wsa:Address>http://www.example.com/MyEndpoint</wsa:Address>
(14)      </wsa:ReplyTo>
(15)      <wsa:To>http://www.example.org/YourEndpoint</wsa:To>
(16)      <ex:MyRefProp xmlns:ex='http://www.example.com/refs' >
(17)          78f2dc229597b529b81c4bef76453c96
(18)      </ex:MyRefProp>
(19)    </s12:Header>
(20)    <s12:Body>
(21)      <wsx:GetPolicy />
(22)    </s12:Body>
(23) </s12:Envelope>
```

Lines (06-08) in Table 1 indicate this is a Get Policy request. Note that Lines (15-18) illustrate a typical pattern where the endpoint is identified by a wsa:To [WS-Addressing] header block (Line 15) as well as an application-specific header block (Lines 16-18). This request is for the policy of the receiver; alternatively, it could include a target namespace within the body (Line 21) to indicate that it is a request for policy within a given target namespace.

Table 2 lists a sample response to the request in Table 1.

Table 2: Sample Get Policy response message.

```
(01) <s12:Envelope
(02)      xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)      xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/03/addressing'
(04)      xmlns:wsp='http://schemas.xmlsoap.org/ws/2002/12/policy'
(05)      xmlns:wsx='http://schemas.xmlsoap.org/ws/2004/03/mex' >
(06)    <s12:Header>
(07)      <wsa:Action>
(08)          http://schemas.xmlsoap.org/ws/2004/03/mex/GetPolicy/Response
(09)      </wsa:Action>
(10)      <wsa:RelatesTo>
(11)          uuid:73d7edfc-5c3c-49b9-ba46-2480caee43e9
(12)      </wsa:RelatesTo>
(13)      <wsa:To>http://www.example.com/MyEndpoint</wsa:To>
(14)    </s12:Header>
(15)    <s12:Body>
(16)      <wsx:GetPolicyResponse>
```

```
(17)       <wsp:Policy
(18)         xmlns:wsse='http://schemas.xmlsoap.org/ws/2002/12/secext' >
(19)         <wsp:OneOrMore>
(20)           <wsse:SecurityToken wsp:Usage='wsp:Required'
(21)                               wsp:Preference='100' >
(22)             <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
(23)           </wsse:SecurityToken>
(24)           <wsse:SecurityToken wsp:Usage='wsp:Required'
(25)                               wsp:Preference='1' >
(26)             <wsse:TokenType>wsse:X509v3</wsse:TokenType>
(27)           </wsse:SecurityToken>
(28)         </wsp:OneOrMore>
(29)       </wsp:Policy>
(30)     </wsx:GetPolicyResponse>
(31)   </s12:Body>
(32) </s12:Envelope>
```

Lines (07-09) in Table 2 indicate this message is a response to a Get Policy request, and Lines (10-12) indicate that it is a response to the request in Table 1. Lines (17-29) contain the policy corresponding to the receiver of the Get Policy request in Table 1.

The corresponding WSDL [WSDL 1.1] for use with other bindings is below. (Refer to Appendix II – WSDL for the complete definition.)

```
<wsdl:message name='GetPolicyMsg' >
  <wsdl:part name='body' element='wsx:GetPolicy' />
</wsdl:message>
<wsdl:message name='GetPolicyResponseMsg' >
  <wsdl:part name='body' element='wsx:GetPolicyResponse' />
</wsdl:message>
<wsdl:message name='PolicyUnavailableFaultMsg' >
  <wsdl:part name='detail'
             element='wsx:PolicyUnavailableFaultDetail' />
</wsdl:message>
<wsdl:message name='UnknownTargetNamespaceFaultMsg' >
  <wsdl:part name='detail'
             element='wsx:UnknownTargetNamespaceFaultDetail' />
</wsdl:message>
<wsdl:message name='InvalidRequestFaultMsg' >
```

```
  <wsdl:part name='detail'
             element='wsx:InvalidRequestFaultDetail' />
</wsdl:message>
<wsdl:message name='ResponseTooLargeFaultMsg' >
  <wsdl:part name='detail'
             element='wsx:PolicyUnavailableFaultDetail' />
</wsdl:message>

<wsdl:portType name='MetadataExchange' >
  <wsdl:operation name='GetPolicy' >
    <wsdl:input message='wsx:GetPolicyMsg' />
    <wsdl:output message='wsx:GetPolicyResponseMsg' />
    <wsdl:fault name='UnknownTargetNamespace'
                message='wsx:UnknownTargetNamespaceFaultMsg' />
    <wsdl:fault name='PolicyUnavailable'
                message='wsx:PolicyUnavailableFaultMsg' />
    <wsdl:fault name='InvalidRequest'
                message='wsx:InvalidRequestFaultMsg' />
    <wsdl:fault name='ResponseTooLarge'
                message='wsx:ResponseTooLargeFaultMsg' />
  </wsdl:operation>
  ...
</wsdl:portType>
```

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 an extension is not recognized it SHOULD be ignored.
- XML namespace prefixes (see Table 3) 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/2004/03/mex
```

Table 3 lists XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

Table 3: 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 [SOAP 1.1]
s12	http://www.w3.org/2003/05/soap-envelope	SOAP 1.2 [SOAP 1.2]
wsa	http://schemas.xmlsoap.org/ws/2004/03/addressing	WS-Addressing [WS-Addressing]
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL [WSDL 1.1]
wsse	http://schemas.xmlsoap.org/ws/2002/12/secext	WS-SecurityPolicy [WS-SecurityPolicy]
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy	WS-Policy [WS-Policy]
wsx	http://schemas.xmlsoap.org/ws/2004/03/mex	This specification
xs	http://www.w3.org/2001/XMLSchema	XML Schema [Part 1, 2]

2.3 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.

3. Retrieving Metadata

3.1 Retrieving Policy

To retrieve the policy [WS-Policy] for a target namespace or an endpoint, a requestor MAY send a Get Policy request message to an endpoint. The normative outline for a Get Policy request is:

```
<s:Envelope ...>
```

```
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/03/mex/GetPolicy/Request
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID> ?
    <wsa:ReplyTo>endpoint-reference</wsa:ReplyTo> ?
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </s:Header>
  <s:Body ...>
    <wsx:GetPolicy ...>
      <wsx:TargetNamespace ...>xs:anyURI</wsx:TargetNamespace> ?
    </wsx:GetPolicy>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/wsa:Action
: MUST be included to define the metadata to be returned. If in addition to this message header a SOAP Action URI is used in the binding for SOAP, the value indicated herein MUST be used for that URI.

/s:Envelope/s:Header/wsa:ReplyTo
: If included, MUST be of type wsa:EndpointReferenceType [WS-Addressing].

/s:Envelope/s:Body/*/wsx:TargetNamespace
: Request is for policy in the indicated target namespace.

/s:Envelope/s:Body/*[not(wsx:TargetNamespace)]
: Request is for policy of the receiver.

Other message information headers defined by WS-Addressing [WS-Addressing] MAY be included in the request and response messages, according to the usage and semantics defined in that specification.

If /s:Envelope/s:Body/*/wsx:TargetNamespace in the corresponding request, and the receiver does not have policy for the specified target namespace, the request MUST fail, and the receiver MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "unknown target namespace"

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wsx:UnknownTargetNamespace
- s12:Reason/s12:Text = e.g., "unknown target namespace"

If /s:Envelope/s:Body/*[not(wsx:TargetNamespace)] in the corresponding request, and the receiver does not expose policy for itself, the request MUST fail, and the receiver MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "policy unavailable for endpoint"

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wsx:PolicyUnavailable
- s12:Reason/s12:Text = e.g., "policy unavailable for endpoint"

If an endpoint accepts a Get Policy request, it MUST reply with a Get Policy response message. The normative outline for a Get Policy response is:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/03/mex/GetPolicy/Response
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo> ?
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </s:Header>
  <s:Body ...>
    <wsx:GetPolicyResponse ...>
    [ <wsp:Policy TargetNamespace='xs:anyURI'? ...>
        ...
      </wsp:Policy> ] +
    </wsx:GetPolicyResponse>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/wsa:Action
: MUST be included in the response message.

/s:Envelope/s:Header/wsa:RelatesTo
: If /s:Envelope/s:Header/wsa:MessageID was included in the corresponding request, this header block MUST be included and MUST have that value.

/s:Envelope/s:Body/*/wsp:Policy
: If /s:Envelope/s:Body/*/wsx:TargetNamespace in the corresponding request, SHOULD be repeated if there is > 1 policy in that target namespace.

/s:Envelope/s:Body/*/wsp:Policy/@TargetNamespace

If /s:Envelope/s:Body/*/wsx:TargetNamespace in the corresponding request, then MUST be the same value.

3.2 Retrieving WSDL

To retrieve the WSDL [WSDL 1.1] for a service, a requestor MAY send a Get WSDL request message to an endpoint. The normative outline for a Get WSDL request is:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/03/mex/GetWSDL/Request
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID> ?
    <wsa:ReplyTo>endpoint-reference</wsa:ReplyTo> ?
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </s:Header>
  <s:Body ...>
    <wsx:GetWSDL ...>
      <wsx:TargetNamespace>xs:anyURI</wsx:TargetNamespace> ?
    </wsx:GetWSDL>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/wsa:Action
: MUST be included to define the metadata to be returned. If in addition to this message header a SOAP Action URI is used in the binding for SOAP, the value indicated herein MUST be used for that URI.

/s:Envelope/s:Header/wsa:ReplyTo
: If included, MUST be of type wsa:EndpointReferenceType [WS-Addressing].

/s:Envelope/s:Body/*/wsx:TargetNamespace
: Request is for WSDL documents in the indicated target namespace.

/s:Envelope/s:Body/*/ [not(wsx:TargetNamespace)]
: Request is for WSDL of the receiver which MUST include exactly one wsdl:service representing the receiver, and MAY include other WSDL constructs.

Other message information headers defined by WS-Addressing [WS-Addressing] MAY be included in the request and response messages, according to the usage and semantics defined in that specification.

If /s:Envelope/s:Body/*/wsx:TargetNamespace, and the receiver does not have WSDL for the specified target namespace, the request MUST fail, and the receiver MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "unknown target namespace"

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wsx:UnknownTargetNamespace
- s12:Reason/s12:Text = e.g., "unknown target namespace"

If /s:Envelope/s:Body/*[not(wsx:TargetNamespace)], and the receiver does not expose WSDL for itself, the request MUST fail, and the receiver MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "WSDL unavailable for endpoint"

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wsx:WSDLUnavailable
- s12:Reason/s12:Text = e.g., "WSDL unavailable for endpoint"

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

If an endpoint accepts a Get WSDL request, it MUST reply with a Get WSDL response message. The normative outline for a Get WSDL response is:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/03/mex/GetWSDL/Response
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo> ?
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </s:Header>
  <s:Body ...>
    <wsx:GetWSDLResponse>
      <wsdl:definitions ...> ... </wsdl:definitions> +
    </wsx:GetWSDLResponse>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/wsa:Action
    MUST be included in the response message.

/s:Envelope/s:Header/wsa:RelatesTo

If /s:Envelope/s:Header/wsa:MessageID was included in the corresponding request, this header block MUST be included and MUST have that value.

/s:Envelope/s:Body/*/wsdl:definitions
If /s:Envelope/s:Body/*/wsx:TargetNamespace in the corresponding request, SHOULD be repeated if there is > 1 WSDL in the target namespace.

/s:Envelope/s:Body/*/wsdl:definitions/@targetNamespace
If /s:Envelope/s:Body/*/wsx:TargetNamespace in the corresponding request, then MUST be the same value.

The need to retrieve specific WSDL files corresponding to particular target namespaces arises in the following scenario: an initial Get WSDL request is used to retrieve the WSDL containing the wsdl:service element corresponding to the endpoint; successive Get WSDL requests are then issued to obtain other WSDL documents on which that definition depends, identified by their target namespace. Other usage scenarios are possible.

The corresponding WSDL [WSDL 1.1] for use with other bindings is below. (Refer to Appendix II – WSDL for the complete definition.)

```
<wsdl:message name='GetWSDLMsg' >
  <wsdl:part name='body' element='wsx:GetWSDL' />
</wsdl:message>
<wsdl:message name='GetWSDLResponseMsg' >
  <wsdl:part name='body' element='wsx:GetWSDLResponse' />
</wsdl:message>
<wsdl:message name='WSDLUnavailableFaultMsg' >
  <wsdl:part name='detail' element='wsx:WSDLUnavailableFaultDetail' />
</wsdl:message>

<wsdl:portType name='MetadataExchange' >
  ...
  <wsdl:operation name='GetWSDL' >
    <wsdl:input message='wsx:GetWSDLMsg' />
    <wsdl:output message='wsx:GetWSDLResponseMsg' />
    <wsdl:fault name='UnknownTargetNamespace'
                message='wsx:UnknownTargetNamespaceFaultMsg' />
    <wsdl:fault name='WSDLUnavailable'
                message='wsx:WSDLUnavailableFaultMsg' />
    <wsdl:fault name='InvalidRequest'
                message='wsx:InvalidRequestFaultMsg' />
    <wsdl:fault name='ResponseTooLarge'
                message='wsx:ResponseTooLargeFaultMsg' />
```

```
  </wsdl:operation>
  ...
</wsdl:portType>
```

Table 4 lists a sample Get WSDL request.

Table 4: Sample Get WSDL request message.

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/03/addressing'
(04)     xmlns:wsx='http://schemas.xmlsoap.org/ws/2004/03/mex' >
(05)   <s12:Header>
(06)     <wsa:Action>
(07)         http://schemas.xmlsoap.org/ws/2004/03/mex/GetWSDL/Request
(08)     </wsa:Action>
(09)     <wsa:MessageID>
(10)         uuid:a6e37bfb-f324-4e71-b33a-4f6d5c6027f4
(11)     </wsa:MessageID>
(12)     <wsa:ReplyTo>
(13)       <wsa:Address>http://www.example.com/MyEndpoint</wsa:Address>
(14)     </wsa:ReplyTo>
(15)     <wsa:To>http://www.example.org/YourEndpoint</wsa:To>
(16)   </s12:Header>
(17)   <s12:Body>
(18)     <wsx:GetWSDL>
(19)       <wsx:TargetNamespace>
(20)           http://www.example.org/stockquote
(21)       </wsx:TargetNamespace>
(22)     </wsx:GetWSDL>
(23)   </s12:Body>
(24) </s12:Envelope>
```

Lines (06-08) in Table 4 indicate this is a Get WSDL request, and Lines (19-21) indicate it is a request for the target namespace "http://www.example.org/stockquote".

Table 5 lists a sample response to the request in Table 4.

Table 5: Sample Get WSDL response message.

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/03/addressing'
(04)     xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
```

```
(05)     xmlns:wsx='http://schemas.xmlsoap.org/ws/2004/03/mex' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)         http://schemas.xmlsoap.org/ws/2004/03/mex/GetWSDL/Response
(09)     </wsa:Action>
(10)     <wsa:RelatesTo>
(11)         uuid:a6e37bfb-f324-4e71-b33a-4f6d5c6027f4
(12)     </wsa:RelatesTo>
(13)     <wsa:To>http://www.example.com/MyEndpoint</wsa:To>
(14)   </s12:Header>
(15)   <s12:Body>
(16)     <wsx:GetWSDLResponse>
(17)       <wsdl:definitions
(18)           name='StockQuote'
(19)           targetNamespace='http://example.com/stockquote'
(20)           xmlns:tns='http://example.com/stockquote'
(21)           xmlns:wsoap='http://schemas.xmlsoap.org/wsdl/soap/' >
(22)         <wsdl:import namespace='http://example.com/stockquote'
(23)                      location='http://example.com/stockquote' />
(24)
(25)         <wsdl:portType name='StockQuotePortType'>
(26)           <wsdl:operation name='GetLastTradePrice'>
(27)             <wsdl:input message='tns:GetLastTradePriceInput'/>
(28)             <wsdl:output message='tns:GetLastTradePriceOutput'/>
(29)           </wsdl:operation>
(30)         </wsdl:portType>
(31)
(32)         <wsdl:service name='StockQuoteService'>
(33)           <wsdl:port name='StockQuotePort'
(34)                      binding='tns:StockQuoteBinding' >
(35)             <wsoap:address
(36)                 location='http://www.example.org/stockquote' />
(37)           </wsdl:port>
(38)         </wsdl:service>
(39)
(40)         <wsdl:service name='StockDetailService'>
```

```
(41)           <wsdl:port name='StockDetailPort'
(42)                      binding='tns:StockDetailBinding' >
(43)             <wsoap:address
(44)                 location='http://www.example.org/stockdetail' />
(45)           </wsdl:port>
(46)         </wsdl:service>
(47)
(48)       </wsdl:definitions>
(49)     </wsx:GetWSDLResponse>
(50)   </s12:Body>
(51) </s12:Envelope>
```

Lines (07-09) in Table 5 indicate this message is a response to a Get WSDL request, and Lines (10-12) indicate that it is a response to the request identified by the message identifier in Table 4. Lines (17-48) contain the WSDL [WSDL 1.1] for the requested target namespace. Note that because a specific target namespace was requested in this case, the WSDL may contain more than one wsdl:service element (Lines 32 and 40). Note that this example includes only a single WSDL; if there is more than one WSDL in the requested target namespace, returning all of them is explicitly encouraged.

3.3 Retrieving Schema

To retrieve an XML Schema [XML Schema Part 1, Part 2], a requestor MAY send a Get Schema request message to an endpoint. The normative outline for a Get Schema request is:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/03/mex/GetSchema/Request
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID> ?
    <wsa:ReplyTo>endpoint-reference</wsa:ReplyTo> ?
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </s:Header>
  <s:Body ...>
    <wsx:GetSchema ...>
      <wsx:TargetNamespace>xs:anyURI</wsx:TargetNamespace>
    </wsx:GetSchema>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/wsa:Action
: MUST be included to define the metadata to be returned. If in addition to this message header a SOAP Action URI is used in the binding for SOAP, the value indicated herein MUST be used for that URI.

/s:Envelope/s:Header/wsa:ReplyTo
: If included, MUST be of type wsa:EndpointReferenceType [WS-Addressing].

/s:Envelope/s:Body/*/wsx:TargetNamespace
: The target namespace of the desired XML Schema.

Other message information headers defined by WS-Addressing [WS-Addressing] MAY be included in the request and response messages, according to the usage and semantics defined in that specification.

If the receiver does not have a schema for the specified target namespace, the request MUST fail, and the receiver MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "unknown target namespace"

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wsx:UnknownTargetNamespace
- s12:Reason/s12:Text = e.g., "unknown target namespace"

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

If an endpoint accepts a Get Schema request, it MUST reply with a Get Schema response message. The normative outline for a Get Schema response is:

```
<s:Envelope ...>
  <s:Header ...>
    <wsa:Action>
        http://schemas.xmlsoap.org/ws/2004/03/mex/GetSchema/Response
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo> ?
    <wsa:To>xs:anyURI</wsa:To>
    ...
  </s:Header>
  <s:Body ...>
    <wsx:GetSchemaResponse>
      <xs:schema ...> ... </xs:schema> +
    </wsx:GetSchemaResponse>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/wsa:Action
MUST be included in the response message.

/s:Envelope/s:Header/wsa:RelatesTo
If /s:Envelope/s:Header/wsa:MessageID was included in the corresponding request, this header block MUST be included and MUST have that value.

/s:Envelope/s:Body/xs:schema
SHOULD be repeated if there is > 1 XML Schema in the target namespace.

/s:Envelope/s:Body/*/xs:schema/@targetNamespace
MUST equal /s:Envelope/s:Body/*/wsx:TargetNamespace in the corresponding request.

In a possible scenario, specific Schema files relevant to the service definition are identified by their target namespaces from the WSDL definition of the service. Other usage scenarios are possible.

The corresponding WSDL [WSDL 1.1] for use with other bindings is below. (Refer to Appendix II – WSDL for the complete definition.)

```
<wsdl:message name='GetSchemaMsg' >
  <wsdl:part name='body' element='wsx:GetSchema' />
</wsdl:message>
<wsdl:message name='GetSchemaResponseMsg' >
  <wsdl:part name='body' element='wsx:GetSchemaResponse' />
</wsdl:message>

<wsdl:portType name='MetadataExchange' >
  ....
  <wsdl:operation name='GetSchema' >
    <wsdl:input message='wsx:GetSchemaMsg' />
    <wsdl:output message='wsx:GetSchemaResponseMsg' />
    <wsdl:fault name='UnknownTargetNamespace'
                message='wsx:UnknownTargetNamespaceFaultMsg' />
    <wsdl:fault name='InvalidRequest'
                message='wsx:InvalidRequestFaultMsg' />
    <wsdl:fault name='ResponseTooLarge'
                message='wsx:ResponseTooLargeFaultMsg' />
  </wsdl:operation>
</wsdl:portType>
```

Table 6 lists a sample Get Schema request message.

Table 6 Sample Get Schema request message.

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/03/addressing'
```

```
(04)     xmlns:wsx='http://schemas.xmlsoap.org/ws/2004/03/mex' >
(05)   <s12:Header>
(06)     <wsa:Action>
(07)         http://schemas.xmlsoap.org/ws/2004/03/mex/GetSchema/Request
(08)     </wsa:Action>
(09)     <wsa:MessageID>
(10)         uuid:25635433-e6d7-42cd-b782-b8fda7d718e0
(11)     </wsa:MessageID>
(12)     <wsa:ReplyTo>
(13)       <wsa:Address>http://www.example.com/MyEndpoint</wsa:Address>
(14)     </wsa:ReplyTo>
(15)     <wsa:To>http://www.example.org/YourEndpoint</wsa:To>
(16)   </s12:Header>
(17)   <s12:Body>
(18)     <wsx:GetSchema>
(19)       <wsx:TargetNamespace>
(20)         http://www.example.org/stockquote
(21)       </wsx:TargetNamespace>
(22)     </wsx:GetSchema>
(23)   </s12:Body>
(24) </s12:Envelope>
```

Lines (06-08) Table 6 indicate this is a Get Schema request, and Lines (19-21) indicate it is a request for the target namespace "http://www.example.org/stockquote".

Table 7 lists a sample response to the request in Table 6.

Table 7: Sample Get Schema response message.

```
(01) <s12:Envelope
(02)     xmlns:s12='http://www.w3.org/2003/05/soap-envelope'
(03)     xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/03/addressing'
(04)     xmlns:wsx='http://schemas.xmlsoap.org/ws/2004/03/mex'
(05)     xmlns:xs='http://www.w3.org/2001/XMLSchema' >
(06)   <s12:Header>
(07)     <wsa:Action>
(08)         http://schemas.xmlsoap.org/ws/2004/03/mex/GetSchema/Response
(09)     </wsa:Action>
(10)     <wsa:RelatesTo>
(11)         uuid:25635433-e6d7-42cd-b782-b8fda7d718e0
(12)     </wsa:RelatesTo>
```

```
(13)     <wsa:To>http://www.example.com/MyEndpoint</wsa:To>
(14)   </s12:Header>
(15)   <s12:Body>
(16)     <wsx:GetSchemaResponse>
(17)       <xs:schema
(18)           targetNamespace='http://example.org/stockquote'
(19)           xmlns='http://www.w3.org/2000/10/XMLSchema'>
(20)         <xs:element name='TradePriceRequest'>
(21)           <xs:complexType>
(22)             <xs:all>
(23)               <xs:element name='tickerSymbol' type='xs:string' />
(24)             </xs:all>
(25)           </xs:complexType>
(26)         </xs:element>
(27)         <xs:element name='TradePrice'>
(28)           <xs:complexType>
(29)             <xs:all>
(30)               <xs:element name='price' type='xs:float'/>
(31)             </xs:all>
(32)           </xs:complexType>
(33)         </xs:element>
(34)       </xs:schema>
(35)     </wsx:GetSchemaResponse>
(36)   </s12:Body>
(37) </s12:Envelope>
```

Lines (07-09) in Table 7 indicate this message is a response to a Get Schema request, and Lines (10-12) indicate that it is a response to the request in Table 6. Lines (17-34) contain the XML Schema [XML Schema Part 1, Part 2] for the requested target namespace. Note that this example includes only a single XML Schema; if there is more than one XML Schema in the requested target namespace, returning them all is explicitly encouraged.

3.4 Other Faults

If a Get Policy, Get WSDL, or Get Schema request message does not comply with the corresponding outline listed above, the request MUST fail, and the receiver MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Client
- faultstring = e.g., "message is invalid"

SOAP 1.2:

- s12:Code/s12:Value = s12:Sender
- s12:Code/s12:Subcode/s12:Value = wsx:InvalidRequest
- s12:Reason/s12:Text = e.g., "message is invalid"

If the amount of data to be sent in a Get Policy response, Get WSDL response, or Get Schema response exceeds what the receiver can include in the response, the request MUST fail, and the receiver MAY generate a SOAP fault as follows:

SOAP 1.1:

- faultcode = s11:Server
- faultstring = e.g., "response too large"

SOAP 1.2:

- s12:Code/s12:Value = s12:Receiver
- s12:Code/s12:Subcode/s12:Value = wsx:ResponseTooLarge
- s12:Reason/s12:Text = e.g., "response too large"

4. Normative Protocol Binding

A binding for the messages described herein to SOAP 1.1 [SOAP 1.1] over HTTP as constrained by the Basic Profile 1.0 [BP 1.0] is RECOMMENDED as a means to bootstrap communication. A Web service is free to support these messages over other bindings in addition to, or in place of, this binding as specified by WSDL [WSDL 1.1], policies, or other mechanisms. In the absence of an explicit specification stating that a different binding must be used, the default SOAP 1.1 over HTTP binding defined here is assumed to apply.

5. Security Considerations

It is strongly RECOMMENDED that the communication between Web services be secured using the mechanisms described in WS-Security [WS-Security, Addendum]. In order to properly secure messages, the body and all relevant headers need to be included in the signature. Specifically, any standard messaging headers, such as those from WS-Addressing [WS-Addressing], need to be signed with the body in order to "bind" the two together.

Different security mechanisms may be desired depending on the frequency of messages. For example, for infrequent messages, public key technologies may be adequate for integrity and confidentiality. However, for high-frequency events, it may be more performant to establish a security context for the events using the mechanisms described in WS-Trust [WS-Trust] and WS-SecureConversation [WS-SecureConversation]. It should be noted that if a shared secret is used it is RECOMMENDED that derived keys be used to strengthen the secret as described in WS-SecureConversation.

Requests for metadata that are not available to anonymous parties are strongly RECOMMENDED to require usage of WS-Security so that the requestor can be authenticated and authorized to access the indicated metadata. Similarly, integrity and confidentiality SHOULD be used whenever metadata has restricted access.

Recipients of metadata are RECOMMENDED to validate the signature to authenticate and verify the integrity of the data. Specifically, recipients SHOULD verify that the sender has the right to "speak" for the metadata. This is important because some metadata,

such as schemas, have embedded target URIs that might be outside the scope of the sender.

Additionally, some metadata formats, such as policies [WS-Policy], may have embedded security semantics. These SHOULD be verified using the same considerations outlined in this section.

The following list summarizes common classes of attacks that apply to this protocol and identifies the mechanism to prevent/mitigate the attacks:

- **Message alteration** – Alteration is prevented by including signatures of the message information using WS-Security.
- **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 [WS-SecurityPolicy]).
- **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 strength 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** – Metadata services are subject to a variety of availability attacks such as application-level denial of service. It is recommended that the mechanisms described in WS-Security be considered as mitigations for some forms of attacks. Other attacks, such as network-level denial of service are harder to avoid. Note that both of these classes attack are outside the scope of this specification.
- **Replay** – Messages may be replayed for a variety of reasons. To detect and eliminate this attack, mechanisms should be used to identify replayed messages such as the timestamp/nonce outlined in WS-Security. Alternatively, and optionally, other technologies, such as sequencing, can also be used to prevent replay of application messages.

6. Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams, including: Luis Felipe Cabrera (Microsoft), Erik Christensen (Microsoft), Timm Falter (SAP), Don Ferguson (IBM), Jeffrey Frey (IBM), Maryann Hondo (IBM), Chris Kaler (Microsoft), Dave Langworthy (Microsoft), Frank Leymann (IBM), Brad Lovering (Microsoft), Steve Millet (Microsoft), Sanjay Patil (SAP), Chris Sharp (IBM), Eugene Sindambiwe (SAP), Tony Storey (IBM), Sara Wong (Microsoft).

7. References

[BP 1.0]

K. Ballinger, et al, "Basic Profile Version 1.0a," August 2003.

[RFC 2119]

S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997.

[SOAP 1.1]

D. Box, et al, "Simple Object Access Protocol (SOAP) 1.1," May 2000.

[SOAP 1.2]
M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework," June 2003.

[WS-Addressing]
A. Bosworth, et al, "Web Services Addressing (WS-Addressing)," March 2004.

[WS-Policy]
D. Box, et al, "Web Services Policy Framework (WS-Policy)," May 2003.

[WS-SecureConversation]
G. Della-Libera, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," December 2002.

[WS-Security]
B. Atkinson, et al, "Web Services Security (WS-Security)," April 2002.

[WS-Security Addendum]
G. Della-Libera, et al, "Web Services Security Addendum," August 2002.

[WS-SecurityPolicy]
G. Della-Libera, et al, "Web Services Security Policy Language (WS-SecurityPolicy)," December 2002.

[WS-Trust]
G. Della-Libera, et al, "Web Services Trust Language (WS-Trust)," December 2002.

[WSDL 1.1]
E. Christensen, et al, "Web Services Description Language (WSDL) 1.1," March 2001.

[XML Schema, Part 1]
H. Thompson, et al, "XML Schema Part 1: Structures," May 2001.

[XML Schema, Part 2]
P. Biron, et al, "XML Schema Part 2: Datatypes," May 2001.

Appendix I – XML Schema

A normative copy of the XML Schema [XML Schema Part 1, Part 2] and WSDL [WSDL 1.1] descriptions 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 description is listed below for convenience.

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

```
<xs:schema
    targetNamespace='http://schemas.xmlsoap.org/ws/2004/03/mex'
    xmlns:tns='http://schemas.xmlsoap.org/ws/2004/03/mex'
    xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/03/addressing'
    xmlns:wsp='http://schemas.xmlsoap.org/ws/2002/12/policy'
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
    elementFormDefault='qualified'
    blockDefault='#all' >
```

```
<xs:import
    namespace='http://schemas.xmlsoap.org/ws/2004/03/addressing'
    schemaLocation='http://schemas.xmlsoap.org/ws/2004/03/addressing' />

<xs:import
    namespace='http://schemas.xmlsoap.org/ws/2002/12/policy'
    schemaLocation='http://schemas.xmlsoap.org/ws/2002/12/Policy/' />

<xs:import
    namespace='http://schemas.xmlsoap.org/wsdl/'
    schemaLocation='http://schemas.xmlsoap.org/wsdl/' />

<xs:import
    namespace='http://www.w3.org/2001/XMLSchema'
    schemaLocation='http://www.w3.org/2001/XMLSchema.xsd' />

<xs:element name='GetPolicy' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='tns:TargetNamespace' minOccurs='0' />
    </xs:sequence>
    <xs:anyAttribute namespace='##any' processContents='lax' />
  </xs:complexType>
</xs:element>
<xs:element name='GetPolicyResponse' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='wsp:Policy' maxOccurs='unbounded' />
    </xs:sequence>
    <xs:anyAttribute namespace='##any' processContents='lax' />
  </xs:complexType>
</xs:element>

<xs:element name='GetWSDL' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='tns:TargetNamespace' minOccurs='0' />
```

```
    </xs:sequence>
    <xs:anyAttribute namespace='##any' processContents='lax' />
  </xs:complexType>
</xs:element>
<xs:element name='GetWSDLResponse' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='wsdl:definitions' maxOccurs='unbounded' />
    </xs:sequence>
    <xs:anyAttribute namespace='##any' processContents='lax' />
  </xs:complexType>
</xs:element>

<xs:element name='GetSchema' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='tns:TargetNamespace' />
    </xs:sequence>
    <xs:anyAttribute namespace='##any' processContents='lax' />
  </xs:complexType>
</xs:element>
<xs:element name='GetSchemaResponse' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='xs:schema' maxOccurs='unbounded' />
    </xs:sequence>
    <xs:anyAttribute namespace='##any' processContents='lax' />
  </xs:complexType>
</xs:element>

<xs:element name='TargetNamespace' type='xs:anyURI' />

<xs:element name='PolicyUnavailableFaultDetail'
            type='tns:OpenDetailType' />
<xs:element name='WSDLUnavailableFaultDetail'
            type='tns:OpenDetailType' />
```

```
  <xs:element name='UnknownTargetNamespaceFaultDetail'
              type='tns:OpenDetailType' />
  <xs:element name='InvalidRequestFaultDetail'
              type='tns:OpenDetailType' />
  <xs:element name='ResponseTooLargeFaultDetail'
              type='tns:OpenDetailType' />

  <xs:complexType name='OpenDetailType' >
    <xs:sequence>
      <xs:any minOccurs='0'
              maxOccurs='unbounded'
              namespace='##any'
              processContents='lax' />
    </xs:sequence>
   <xs:anyAttribute namespace='##any'
                    processContents='lax' />
  </xs:complexType>

</xs:schema>
```

Appendix II – WSDL

A non-normative copy of the WSDL [WSDL 1.1] description for this specification is listed below for convenience.

```
<wsdl:definitions
    targetNamespace='http://schemas.xmlsoap.org/ws/2004/03/mex'
    xmlns:wsx='http://schemas.xmlsoap.org/ws/2004/03/mex'
    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/2004/03/mex'
        schemaLocation='metadataexchange.xsd' />
    </xs:schema>
  </wsdl:types>

  <wsdl:message name='GetPolicyMsg' >
```

```
  <wsdl:part name='body' element='wsx:GetPolicy' />
</wsdl:message>
<wsdl:message name='GetPolicyResponseMsg' >
  <wsdl:part name='body' element='wsx:GetPolicyResponse' />
</wsdl:message>

<wsdl:message name='PolicyUnavailableFaultMsg' >
  <wsdl:part name='detail'
             element='wsx:PolicyUnavailableFaultDetail' />
</wsdl:message>
<wsdl:message name='UnknownTargetNamespaceFaultMsg' >
  <wsdl:part name='detail'
             element='wsx:UnknownTargetNamespaceFaultDetail' />
</wsdl:message>
<wsdl:message name='InvalidRequestFaultMsg' >
  <wsdl:part name='detail'
             element='wsx:InvalidRequestFaultDetail' />
</wsdl:message>
<wsdl:message name='ResponseTooLargeFaultMsg' >
  <wsdl:part name='detail'
             element='wsx:PolicyUnavailableFaultDetail' />
</wsdl:message>

<wsdl:message name='GetWSDLMsg' >
  <wsdl:part name='body' element='wsx:GetWSDL' />
</wsdl:message>
<wsdl:message name='GetWSDLResponseMsg' >
  <wsdl:part name='body' element='wsx:GetWSDLResponse' />
</wsdl:message>

<wsdl:message name='WSDLUnavailableFaultMsg' >
  <wsdl:part name='detail'
             element='wsx:WSDLUnavailableFaultDetail' />
</wsdl:message>

<wsdl:message name='GetSchemaMsg' >
  <wsdl:part name='body' element='wsx:GetSchema' />
```

```
</wsdl:message>
<wsdl:message name='GetSchemaResponseMsg' >
  <wsdl:part name='body' element='wsx:GetSchemaResponse' />
</wsdl:message>

<wsdl:portType name='MetadataExchange' >
  <wsdl:operation name='GetPolicy' >
    <wsdl:input message='wsx:GetPolicyMsg' />
    <wsdl:output message='wsx:GetPolicyResponseMsg' />
    <wsdl:fault name='UnknownTargetNamespace'
                message='wsx:UnknownTargetNamespaceFaultMsg' />
    <wsdl:fault name='PolicyUnavailable'
                message='wsx:PolicyUnavailableFaultMsg' />
    <wsdl:fault name='InvalidRequest'
                message='wsx:InvalidRequestFaultMsg' />
    <wsdl:fault name='ResponseTooLarge'
                message='wsx:ResponseTooLargeFaultMsg' />
  </wsdl:operation>
  <wsdl:operation name='GetWSDL' >
    <wsdl:input message='wsx:GetWSDLMsg' />
    <wsdl:output message='wsx:GetWSDLResponseMsg' />
    <wsdl:fault name='UnknownTargetNamespace'
                message='wsx:UnknownTargetNamespaceFaultMsg' />
    <wsdl:fault name='WSDLUnavailable'
                message='wsx:WSDLUnavailableFaultMsg' />
    <wsdl:fault name='InvalidRequest'
                message='wsx:InvalidRequestFaultMsg' />
    <wsdl:fault name='ResponseTooLarge'
                message='wsx:ResponseTooLargeFaultMsg' />
  </wsdl:operation>
  <wsdl:operation name='GetSchema' >
    <wsdl:input message='wsx:GetSchemaMsg' />
    <wsdl:output message='wsx:GetSchemaResponseMsg' />
    <wsdl:fault name='UnknownTargetNamespace'
                message='wsx:UnknownTargetNamespaceFaultMsg' />
    <wsdl:fault name='InvalidRequest'
                message='wsx:InvalidRequestFaultMsg' />
```

```
      <wsdl:fault name='ResponseTooLarge'
                  message='wsx:ResponseTooLargeFaultMsg' />
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>
```