WS-MessageDelivery Version 1.0

W3C Member Submission 26 April 2004

This version:
http://www.w3.org/Submission/2004/SUBM-ws-messagedelivery-20040426/
Latest version:
http://www.w3.org/Submission/ws-messagedelivery/
Authors:
Anish Karmarkar, Oracle Corporation (Editor)
Ümit Yalçinalp, Oracle Corporation (Editor)
Mark Hapner, Sun Microsystems Inc.
Frederick Hirsch, Nokia Corporation
Dave Ingham, Arjuna Technologies Limited
Mark Little, Arjuna Technologies Limited
Michael Mahan, Nokia Corporation
Jeff Mischkinsky, Oracle Corporation
Dale Moberg, Cyclone Commerce Inc.
Eric Newcomer, IONA Technologies
Steve Ross-Talbot, Enigmatec Corporation
Pete Wenzel, SeeBeyond Technology Corporation

This document is also available in these non-normative formats: PDF.

Abstract

WS-MessageDelivery specification defines a mechanism to reference Web services *(WSRef)*, essential *abstract message delivery properties* (AMDP), a SOAP binding for those properties, and the relationship of those properties to WSDL definitions and message exchange patterns. These properties enable SOAP messages to be transport independent - extending messaging capability to use separate transport protocol sessions or even using different transport protocols within the context of a message exchange pattern (MEP). Message delivery details are surfaced to the application layer, extending SOAP processors to use a wider range of message patterns and transport protocols to accomplish a Web service interaction. The abstract message delivery properties include web service references, message identification and message references. This specification outlines in detail how to build message exchange patterns consistent with

WSDL 1.1 or WSDL 2.0 using the definitions in the specification. The semantics and mapping for the *Callback Pattern*, a commonly used message exchange pattern as a composite pattern, is defined. The Web service References (WSRef), Abstract Message Delivery Properties and a SOAP binding are designed for interoperability and extensibility.

Status of this Document

By publishing this document, W3C acknowledges that Arjuna Technologies Limited, Cyclone Commerce Inc., Enigmatec Corporation, IONA Technologies, Nokia Corporation, Oracle Corporation, SeeBeyond Technology Corporation, Sun Microsystems Inc. have made a formal submission to W3C for discussion. Publication of this document by W3C indicates no endorsement of its content by W3C, nor that W3C has, is, or will be allocating any resources to the issues addressed by it. This document is not the product of a chartered W3C group, but is published as potential input to the W3C Process. Publication of acknowledged Member Submissions at the W3C site is one of the benefits of W3C Membership. Please consult the requirements associated with Member Submissions of section 3.3 of the W3C Patent Policy. Please consult the complete list of acknowledged W3C Member Submissions.

Table of Contents

Appendices

1 Introduction

Message delivery is fundamental to all aspects of Web services, making the message delivery properties of targeting of Web services, message identification and message referencing essential. This is especially important when building various message exchange patterns such as a callback pattern. The specification of message delivery properties must take into account how WSDL definitions are used to create message exchange patterns and enable and leverage such usage. At the same time, such specifications must be minimal to support interoperability at this core of Web services. This specification defines a Web service reference, Abstract Message Delivery Properties, a SOAP binding, and the relationship to WSDL without introducing additional information that may hinder interoperability. The schemas are designed to be extensible, but the core definition is minimal.

A WSDL document defines the exchange of messages that enable the interaction with a Web service. There are four Message Exchange Patterns (MEP) defined in a WSDL 1.1 document and they constitute the building blocks for specifying complex interactions. The Message Exchange Patterns, referred as transmission primitives, in WSDL 1.1 are *implicit* and suggested by the conventions used by the operations that utilize them. The WSDL 2.0 specification, which is currently in progress, will formally specify a set of MEPs and will allow additional patterns to be defined and used.

As Web services are deployed for non-trivial business applications involving MEPs, it is necessary to address the problems of delivering a message to a node that participates in an MEP. These include responding asynchronously to requests, correlating messages to enable an MEP and referring to a Web service in a message in a transport independent manner.

It is important to allow different transport sessions or even different transport protocols to be used for separate paths of a message exchange pattern, and the abstract message delivery properties outlined in this specification support this usage. These properties surface message binding information to the SOAP application layer, allowing a Web service greater flexibility to choose the appropriate transport that meets business requirements. For instance, in the mobile environment, devices are typically not HTTP addressable and this is problematic for use cases involving asynchronous messaging or events. This specification provides a mechanism to pass the correct mobile binding information to the SOAP node that forwards the message to the mobile device. A detailed use case for message delivery with mobile applications is covered in Appendix C **C Message Delivery in a Mobile Context** [p.53] .

Complex business applications require support for notification, asynchronous request-response, and long-lived conversations within a context. This requires that messages that are sent to/from a Web service be allowed to identify the Web service or a client that is the intended destination/source for the message. Further, messages may travel over multiple links possibly over multiple transports and may be acted upon asynchronously. Therefore, it is necessary to identify a set of properties that are independent of the transport that enable message delivery in the context of Web services.

This document specifies an abstract set of message delivery properties that enable message delivery for Web services that utilize Message Exchange Patterns associated with WSDL documents. We show how these properties apply to MEPs defined in WSDL 1.1 documents each time a message is exchanged in a specific direction. We also illustrate how they enable a common message exchange pattern, *Callback (CB) pattern*, defined as a composite pattern in this document. A composite pattern is a message exchange pattern that is composed of one or more well defined patterns in WSDL that define a logical unit of

exchange.

Although the focus of this document is enabling message delivery for Web services that use WSDL 1.1 descriptions, the properties and techniques introduced here are also applicable in the future. They are shown to apply within the context of additional MEPs currently being defined by the WSDL 2.0 specification.

This document defines:

1. Web service References (WSRef)
2. Abstract Message Delivery Properties (AMDP) and their mapping to SOAP 1.1 and SOAP 1.2
3. Use of AMDP in the context of MEPs defined in WSDL 1.1
4. Callback pattern implementation using AMDP
5. Use of AMDP in the context of MEPs in defined in WSDL 2.0

1.1 Example

Consider a request-response operation description in a WSDL 1.1 document. Typically, when used with SOAP over HTTP, the same HTTP connection is used to send the request and response messages and request/response correlation is implicit. This implicit approach does not work well for asynchronous responses or when the message goes through multiple hops or when the request and response message use different transports. The example SOAP messages show how AMDP can be used in the request and response messages to implement the request-response operation without relying on transport specific features.

```
Request Message:

<soap11:Envelope
    xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery"
    xmlns:wsdl11="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/">
<soap11:Header xmlns:myns="http://example.com/wsdl">
<wsmd:MessageOriginator>
    <wsmd:uri>http://example.org/client-A </wsmd:uri>
</wsmd:MessageOriginator>

<wsmd:MessageDestination
    wsmd:wsdlLocation="http://example.com/wsdl http://example.com/wsdl/app.wsdl"
    wsmd:targetNamespace="http://example.com/wsdl">
    <wsdl11:service name="myservice" wsmd:portType="myns:myPortType">
        <wsdl11:port name="myport" binding="myns:myBinding">
            <soapbind:address location="http://example.com/wsdl/impl"/>
        </wsdl11:port>
    </wsdl11:service>
</wsmd:MessageDestination>

<wsmd:ReplyDestination>
```

```
    <wsmd:uri>http://example.org/client-A/replyDestination</wsmd:uri>
</wsmd:ReplyDestination>

<wsmd:FaultDestination>
    <wsmd:uri>http://example.org/client-A/faultDestination</wsmd:uri>
</wsmd:FaultDestination>

<wsmd:MessageID>
    uuid:58f202ac-22cf-11d1-b12d-002035b29092
</wsmd:MessageID>

<wsmd:OperationName>myRequestResponseOperation</wsmd:OperationName>
...
</soap11:Header>
<soap11:Body>
...
</soap11:Body>
</soap11:Envelope>
```

There are six SOAP header blocks in the request message:

1. `wsmd:MessageOriginator` - This header block specifies the identity of the client invoking the Web service.
2. `wsmd:MessageDestination` - This header block identifies the service element in the WSDL document that described the service to which the request message is being sent.
3. `wsmd:ReplyDestination` - This header block identifies the dereferenceable URI that is used by the Web service to send the response message.
4. `wsmd:FaultDestination` - This header block identifies the dereferenceable URI that is used by the Web service to send a fault back.
5. `wsmd:MessageID` - This header block assigns a unique URI to the message and is used to identify the message. The unique URI in this header is used for correlating the response message with the request message.
6. `wsmd:OperationName` - This header block identifies the name of the operation that designates the specific message exchange.

```
Response Message:

<soap11:Envelope
     xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery"
     xmlns:wsdl11="http://schemas.xmlsoap.org/wsdl/"
     xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/">
<soap11:Header xmlns:myns="http://example.com/wsdl">
<wsmd:MessageOriginator
   wsmd:wsdlLocation="http://example.com/wsdl http://example.com/wsdl/app.wsdl"
   wsmd:targetNamespace="http://example.com/wsdl">
   <wsdl11:service name="myservice" wsmd:portType="myns:myPortType">
      <wsdl11:port name="myport" binding="myns:myBinding">
         <soapbind:address location="http://example.com/wsdl/impl"/>
```

```
      </wsdl11:port>
   </wsdl11:service>
</wsmd:MessageOriginator>

<wsmd:MessageDestination>
   <wsmd:uri>http://example.com/Client-A/replyDestination</wsmd:uri>
</wsmd:MessageDestination>

<wsmd:MessageID>
   uuid:5a389ad2-22dd-11d1-aa77-002035b29092
</wsmd:MessageID>

<wsmd:MessageReference
   wsmd:reason="http://www.w3.org/2004/04/ws-messagedelivery/reason/response">
   uuid:58f202ac-22cf-11d1-b12d-002035b29092
</wsmd:MessageReference>

<wsmd:OperationName>myRequestResponseOperation</wsmd:OperationName>
...
</soap11:Header>
<soap11:Body>...
</soap11:Body>
</soap11:Envelope>
```

There are five SOAP header blocks in the response message:

1. `wsmd:MessageOriginator` - This header block specifies the service that sent the response message. This value is used to determine the sender of the response message.

2. `wsmd:MessageDestination` - This header block identifies the sender of the initial request.

3. `wsmd:MessageID` - This header block assigns a unique URI to the message and is used to identify the message.

4. `wsmd:MessageReference` - This header block identifies the request message to which this response is being sent. The attribute `wsmd:reason` specifies that the message being sent is a response in a request-response operation. This header is used to correlate the response message with the request message at the initial sender.

5. `wsmd:OperationName` - This header block identifies the name of the operation that designates the specific message exchange.

The advantage of using the SOAP header blocks is that, all the information is in the SOAP message and is transport independent. For example, `myns:myBinding` could use an asynchronous transport that does not provide direct support for correlating request and response SOAP messages (unlike HTTP). This would allow the implementation to use such an asynchronous transport binding to implement the request-response operation.

1.2 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 [IETF RFC 2119] [p.40] .

This specification uses a number of namespace prefixes throughout; they are listed in Table 1. Note that the choice of any namespace prefix is arbitrary and not semantically significant [XML Information Set] [p.40] .

Table 1. Namespace-Prefix Binding used in this specification

Prefix	Namespace
wsdl11	"http://schemas.xmlsoap.org/wsdl/"
soap11	"http://schemas.xmlsoap.org/soap/envelope/"
soap12	"http://www.w3.org/2003/05/soap-envelope"
wsmd	"http://www.w3.org/2004/04/ws-messagedelivery"
xs	"http://www.w3.org/2001/XMLSchema"

1.3 Conformance

This specification describes AMDP, mapping of AMDP to SOAP, use of AMDP in MEPs, description of WS-MessageDelivery feature in WSDL, WSRefs and the Callback pattern. This specification does not mandate a particular implementation, but all the "MUST" and "MUST NOT" must be correctly implemented by an implementation that claims conformance to this specification. Typically, whether a Web service adheres to this specification or not is expressed in the WSDL document that describes the Web service through the use of extensibility points and features, as described in this specification.

An implementation is not required to implement all the mandatory aspects of this specification. For example, an implementation that does not implement the Callback pattern does not have to implement the requirements specified for Callback.

1.4 Relation to Other Specifications

This specification is designed to work in conjunction with existing XML technologies, including but not limited to SOAP 1.1/1.2 and WSDL 1.1. WS-MessageDelivery can be used as a building block for other technologies. For example, other specifications can use WSRefs in message payloads to implement complex services. This specification can be used to implement MEPs defined in WSDL 1.1; especially for message that require end-to-end semantics and for asynchronous transports.

The ideas in this specification could also be used with the current Working Group Draft of WSDL 2.0 [WSDL 2.0 Part 1] [p.40] and the MEPs defined therein [WSDL 2.0 Part 2] [p.40] . Sections **6 Message Delivery and WSDL 2.0** [p.37] and **B Appendix: WSDL 2.0 and AMDPs** [p.44] provide an illustration

of how this might occur.

2 Web service Reference

Message delivery requires a mechanism to reference a Web service. A Web service reference (WSRef) is used to identify and reference a Web service. A WSRef contains enough information to get to the Web service contract containing the message formats, bindings, endpoints and other meta-information needed to dereference a WSRef.

2.1 WSRefs in WSDL 1.1

A WSDL 1.1 Web service reference is considered to be an element whose type is ultimately derived from `wsdl11:tService`, with the following restrictions:

- All `wsdl11:port` children element of the WSRef MUST implement the same `wsdl11:portType`. This `wsdl11:portType` may optionally be indicated as a value of the attribute '`wsmd:portType`' as described below.
- Bindings for all `wsdl11:port` children elements must bind every `wsdl11:part` of the input or output `wsdl11:message` in the `wsdl11:portType` to the corresponding input or output protocol elements that are exchanged over the network. This restriction is very similar to R2209 of WS-I Basic Profile 1.0 [BP 1.0] [p.40] .

The restrictions listed above ensure that all the ports within a WSRef implement the same contract (i.e. the portType).

This definition of a WSRef provides an extensible, WSDL 1.1 centric, typed notion of a Web service reference. It also makes the `wsdl11:service` element a WSRef (when the restrictions listed above are also met). Using XML Schema derivation by restriction, it is possible to declare a WSRef and fix the portType, binding or both. Examples of such WSRefs with fixed portType/binding are listed in section **2.1.2 Examples** [p.12] .

A WSRef, similar to a `service` element in a WSDL description, may contain assertions about the capability of the service that it represents, such as QoS assertions. They are expected to be expressed using extensibility elements within the `wsdl11:service` element.

2.1.1 wsmd:portType attribute

The `wsmd:portType` attribute identifies the `wsdl11:portType` of all the ports defined within a `wsdl11:service` element or an element ultimately derived from `wsdl11:tService`. This attribute is used since WSDL 1.1 does not restrict a service to a single portType.

The type of the `wsmd:portType` *attribute information item* is *xs:QName* and identifies a WSDL 1.1 portType.

The portType *attribute information item* has the following Infoset properties:

- A [local name] of `portType`.
- A [namespace name] of "http://www.w3.org/2004/04/ws-messagedelivery".

2.1.2 Examples

```
<wsdl11:definitions
     targetNamespace="http://example.com/wsdl11-example"
     xmlns:tns="http://example.com/wsdl11-example"
     xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery"
     xmlns:wsdl11="http://schemas.xmlsoap.org/wsdl/">
. . .
<wsdl11:service name="myservice" wsmd:portType="tns:myportType">
   <wsdl11:port name="myport" binding="tns:mybinding">
       <soapbind:address location="http://example.com/wsdl-example1/impl"/>
   </wsdl11:port>
</wsdl11:service>
</wsdl:definitions>
```

The example above specifies a WSDL 1.1 `service` element as a WSRef. The attribute `wsmd:portType` identifies the portType of all the ports within the `service` element.

It is also possible to use XML schema to create a WSDL 1.1 WSRef.

There are three use cases for the use of schema to specify WSRefs: static, semi-static and dynamic.

1. In the static case, the portType and the binding information of the WSRef is known before hand and this information is incorporated into the schema of the web service as shown in the example below.

```
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://example.com/wsdl11-static"
    targetNamespace="http://example.com/wsdl11-static" >
<xs:import namespace="http://schemas.xmlsoap.org/wsdl/"/>
. . .
<xs:element name="StaticReference"type="tns:StaticReferenceType"/>
<xs:complexType name="StaticReferenceType">
    <xs:complexContent>
        <xs:restriction base="wsdl11:tService>
            <xs:sequence >
                <xs:element name='port' type='tns:restrictedPort'/>
                <xs:any namespace='##other' processContents='lax'/>
            </xs:sequence>
            <xs:attribute name='name' type='xs:NCName' use='required'/>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="restrictedPort">
    <xs:complexContent>
         <xs:restriction base="wsdl11:tPort">
             <xs:attribute name="name" type="xs:NCName" use="required"/>          <xs:attribute name='binding' type='xs:QName' use='required'
                          fixed="tns:bindingA"/>
         </xs:restriction></xs:complexContent>
</xs:complexType>
</xs:schema>
```

2. In the semi-static case, the portType of the WSRef is known before hand but not the binding. Hence, only the portType is indicated in the schema as shown below.

```
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://example.com/wsdl11-semi-static"
    xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery"
    targetNamespace="http://example.com/wsdl11-semi-static" >
<xs:import namespace="http://schemas.xmlsoap.org/wsdl/"/>
. . .
<xs:element name="SemiStaticReference"type="tns:SemiStaticReferenceType"/>
<xs:complexType name="SemiStaticReferenceType">
    <xs:complexContent>
        <xs:restriction base="wsdl11:tService>
```

```
            <xs:sequence>
                <xs:element name='port' type='wsdl11:tPort'/>
            </xs:sequence>
            <xs:attribute name='name' type='xs:NCName' use='required'/>          <xs:attribute name="portType" type="xs:QName"use="required"
                        fixed="tns:portTypeA"/>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
</xs:schema>
```

3. In the dynamic case, the portType and the binding of the WSRef is not known before, as shown in the example below.

```
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://example.com/wsdl11-dynamic"
    xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery"
    targetNamespace="http://example.com/wsdl11-dynamic" >
<xs:import namespace="http://schemas.xmlsoap.org/wsdl/"/>
. . .<xs:element name="DynamicReference" type="wsdl11:tService"/>
</xs:schema>
```

3 Abstract Message Delivery Properties

This section defines abstract message delivery properties (AMDP) that enable message delivery for Web services. These abstract properties are not part of the message definitions that describe input and/or output and/or fault messages in WSDL. They are additional properties that enable message delivery.

First, the *wsmd:destination* type is defined below (see **3.1 wsmd:destination type** [p.14]). This type is used to define several of the AMDPs.

There are seven properties that constitute AMDP:

1. MessageOriginator - can be used to dynamically specify the sender of a message (see **3.2 MessageOriginator** [p.17]).
2. MessageDestination - can be used to dynamically specify the destination of a message (see **3.3 MessageDestination** [p.18])
3. ReplyDestination - can be used to dynamically specify the destination to which a reply may be sent (see **3.4 ReplyDestination** [p.19])
4. FaultDestination - can be used to dynamically specify the destination to which a fault may be sent (see **3.5 FaultDestination** [p.19])
5. MessageID - can be used to uniquely identify a message (see **3.6 MessageID** [p.20])
6. MessageReference - can be used to specify relationships between messages (see **3.7 MessageReference** [p.20])
7. OperationName - can be used to indicate the operation that indicates the message exchange (see **3.8 OperationName** [p.22])

3.1 *wsmd:destination type*

The *wsmd:destination* designates a destination for messages. The destinations significant for message delivery are represented by *wsmd:destination* as shown by the following pseudo-schema syntax:

```
<... wsmd:targetNamespace="xs:anyURI"?
     wsmd:wsdlLocation="list-of-xs:anyURI"? >
[<wsdl11:service> | <wsmd:uri> | <wsmd:serviceQName> | <extensibility_element>* ]</...>
```

A *wsmd:destination* type MUST contains one of the following

1. a URI or
2. a QName
3. a WSDL `service` element

It allows two optional attributes called `wsmd:wsdlLocation` and `wsmd:targetNamespace` . These attributes are defined below in detail.

When an element of type *wsmd:destination* contains a URI, this URI identifies the destination which is either not a Web service, or a web service that shares the binding and the message that is being specified by the WSDL definition. For example, a client invoking a request-response operation of a Web service would be represented by a URI since the client agrees to use the binding that is defined by the service.

The *wsmd:destination* may contain a WSDL 1.1 `service` element. In the case where *wsmd:destination* contains a `service` element, the destination designates a Web service. The `service` elements as noted above are WSRefs. This type can be used in conjunction with WSDL 1.1 descriptions.

Note:

A *wsmd:destination* may contain a WSDL 2.0 service element. Please refer to **6 Message Delivery and WSDL 2.0** [p.37] for discussion of how AMDPs may apply to Web services that utilize WSDL 2.0 descriptions.

The *wsmd:destination* may also contain a QName that corresponds to a service element defined in a WSDL document.

The *wsmd:destination* type is extensible, and hence may contain additional elements or attributes that are not defined using the `wsmd` target namespace. The properties MessageOriginator, MessageDestination, ReplyDestination, and FaultDestination are represented by using *wsmd:destination* schema type. See **A Appendix: Schema** [p.42] for full description.

The attribute value `wsmd:wsdlLocation` is ignored when a destination is designated by a URI. The attribute value `wsmd:targetNamespace` is ignored when a destination is specified using a service QName or a URI.

3.1.1 `wsdlLocation` *attribute information item*

The attribute `wsmd:wsdlLocation` points to the location(s) of the WSDL document(s) in which the service element and/or its dependent components description are located. This attribute has a syntax and semantics similar to the XML Schema `xs:schemaLocation` attribute. It is RECOMMENDED that this attribute be present when a `service` element is used as a destination (instead of a URI). This attribute MUST NOT be present when the destination is a URI or a service QName. This attribute can also be used on elements that are WSRefs but not of type *wsmd:destination*.

The wsdlLocation *attribute information item* has the following Infoset properties:

- A [local name] of `wsdlLocation`
- A [namespace name] which has a value of "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the `wsdlLocation` *attribute information item* is a list of *xs:anyURI* pair. The first item in the pair identifies the WSDL targetNamespace and the second item in the pair specifies a dereferenceable URI where the WSDL document that defines the targetNamespace is located.

For a usage example refer to **1.1 Example** [p.6] .

Note:

The recent draft of WSDL 2.0 specification [WSDL 2.0 Part 1] [p.40] also defines a global attribute `wsdlLocation` in the namespace "http://www.w3.org/2004/03/wsdl-instance" that is very similar to this definition. At present, the specification is not clear whether the use of this attribute is specific to definitions using WSDL 2.0 only, hence AMDP definitions in this document retain the `wsmd:wsdlLocation` attribute that can be used for all versions of WSDL documents. However, the `wsdli:wsdlLocation` can be used interchangeably with `wsmd:wsdlLocation`.

3.1.2 `targetNamespace` *attribute information item*

The attribute `wsmd:targetNamespace` specifies the targetNamespace of the WSDL 1.1 `service` element. WSDL 1.1 does not require WSDL definitions to have target namespaces. This attribute therefore MUST be specified when the destination is a WSDL 1.1 `service` element and the `service` element does have a target namespace.

This attribute can also be used on elements that are WSRefs but not of type *wsmd:destination*. This attribute MUST NOT be present when the destination is a URI.

The `targetNamespace` *attribute information item* has the following Infoset properties:

- A [local name] of `targetNamespace`
- A [namespace name] which has a value of "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the `targetNamespace` *attribute information item* is *xs:anyURI*.

For a usage example refer to **1.1 Example** [p.6]

3.1.3 `uri` *element information item*

The `uri` *element information item* has the following Infoset properties:

- A [local name] of `uri`
- A [namespace name] which has a value of "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the uri *element information item* is *xs:anyURI*.

This *element information item* is used to represent nodes that participate in a MEP but may not be Web services themselves (i.e., clients). A special URI value "http://www.w3.org/2004/04/ws-messagedelivery/destination/transport-specified" MAY be used to indicate destinations that either do not have a WSDL service description (such as Web service clients) or destinations that do not have a dereferenceable endpoint. The underlying transport mechanisms, such as HTTP connections, may be used to distinguish such destinations.

The URI "http://www.w3.org/2004/04/ws-messagedelivery/destination/transport-specified" does not uniquely identify a specific node. When used, the destination MUST be identified by using transport specific mechanisms.

3.1.4 `serviceQName` *element information item*

The `serviceQName` *element information item* has the following Infoset properties:

- A [local name] of `serviceQName`
- A [namespace name] which has a value of "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the uri *element information item* is *xs:QName*.

This *element information item* is used to represent the QName of a WSDL service element defined in a WSDL document. A QName is used in conjunction with a WSDL description that specifies the service. It is recommended that a QName be used only when the WSDL description containing the service element utilizing the QName is cached by the parties that participate in the message exchange.

3.2 MessageOriginator

The MessageOriginator property identifies the node that sent the message. MessageOriginator is defined by the following pseudo-schema:

```
<wsmd:MessageOriginator
    wsmd:targetNamespace="xs:anyURI"?
    wsmd:wsdlLocation="list-of-xs:anyURI"? >
[<wsdl11:service> | <wsmd:uri> | <wsmd:serviceQName> | <extensibility_element>*]
</wsmd:MessageOriginator>
```

The `MessageOriginator` *element information item* has the following Infoset properties:

- A [local name] of `MessageOriginator`.
- A [namespace name] of "http://www.w3.org/2004/04/ws-messagedelivery".
- Zero or more *attribute information item*s amongst its [attributes] as follows:
  - An OPTIONAL `wsmd:targetNamespace` *attribute information item* as described above
  - An OPTIONAL `wsmd:wsdlLocation` *attribute information item* as described above.
  - Zero or more namespace qualified *attribute information item*s. The [namespace name] of such *attribute information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".
- One or more *element information item*s as follows:

- Either a `service` *element information item* as defined by WSDL 1.1, a `uri` *element information item* or a `serviceQName` *element information item* as defined above.
- Zero or more namespace-qualified *element information item*s amongst its [children]. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the `MessageOriginator` *element information item* is *wsmd:destination*.

3.3 MessageDestination

MessageDestination identifies the destination or target of the message. MessageDestination is defined by the following pseudo-schema:

```
<wsmd:MessageDestination
    wsmd:targetNamespace="xs:anyURI"?
    wsmd:wsdlLocation="list-of-xs:anyURI"? >
[<wsdl11:service> | <wsmd:uri> | <wsmd:serviceQName> | <extensibility_element>*]
</wsmd:MessageDestination>
```

The `MessageDestination` *element information item* has the following Infoset properties:

- A [local name] of `MessageDestination`.
- A [namespace name] of "http://www.w3.org/2004/04/ws-messagedelivery".
- Zero or more *attribute information item*s amongst its [attributes] as follows:
  - An OPTIONAL `wsmd:targetNamespace` *attribute information item* as described above
  - An OPTIONAL `wsmd:wsdlLocation` *attribute information item* as described above.
  - Zero or more namespace qualified *attribute information item*s. The [namespace name] of such *attribute information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".
- One or more *element information item*s as follows:
  - Either a `service` *element information item* as defined by WSDL 1.1, a `uri` *element information item* or a `serviceQName` *element information item* as defined above.
  - Zero or more namespace-qualified *element information item*s amongst its [children]. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the `MessageDestination` *element information item* is *wsmd:destination*.

3.4 ReplyDestination

ReplyDestination identifies the destination to which a reply to the message may be sent. The value of this property, if present, MUST be used to send a reply when the binding used does not specify how and where the reply is sent. ReplyDestination is defined by the following pseudo-schema:

```
<wsmd:ReplyDestination
    wsmd:targetNamespace="xs:anyURI"?
    wsmd:wsdlLocation="list-of-xs:anyURI"? >
[<wsdl11:service> | <wsmd:uri> | <wsmd:serviceQName> | <extensibility_element>*]
</wsmd:ReplyDestination>
```

The `ReplyDestination` *element information item* has the following Infoset properties:

- A [local name] of `ReplyDestination`.
- A [namespace name] of "http://www.w3.org/2004/04/ws-messagedelivery".
- Zero or more *attribute information item*s amongst its [attributes] as follows:
    - An OPTIONAL `wsmd:targetNamespace` *attribute information item* as described above
    - An OPTIONAL `wsmd:wsdlLocation` *attribute information item* as described above.
    - Zero or more namespace qualified *attribute information item*s. The [namespace name] of such *attribute information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".
- One or more *element information item*s as follows:
    - Either a `service` *element information item* as defined by WSDL 1.1, a `uri` *element information item* or a `serviceQName` *element information item* as defined above.
    - Zero or more namespace-qualified *element information item*s amongst its [children]. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the `ReplyDestination` *element information item* is *wsmd:destination.*

3.5 FaultDestination

FaultDestination identifies the destination to which a fault may be sent. The value of this property, if present, MUST be used to send a fault back when the binding used does not specify how and where the reply is sent.

FaultDestination is defined by the following pseudo-schema:

```
<wsmd:FaultDestination
    wsmd:targetNamespace="xs:anyURI"?
    wsmd:wsdlLocation="list-of-xs:anyURI"? >
[<wsdl11:service> | <wsmd:uri> | <wsmd:serviceQName> | <extensibility_element>*]
</wsmd:FaultDestination>
```

The `FaultDestination` *element information item* has the following Infoset properties:

- A [local name] of `FaultDestination`.
- A [namespace name] of "http://www.w3.org/2004/04/ws-messagedelivery".
- Zero or more *attribute information item*s amongst its [attributes] as follows:
  - An OPTIONAL `wsmd:targetNamespace` *attribute information item* as described above
  - An OPTIONAL `wsmd:wsdlLocation` *attribute information item* as described above.
  - Zero or more namespace qualified *attribute information item*s. The [namespace name] of such *attribute information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".
- One or more *element information item*s as follows:
  - Either a `service` *element information item* as defined by WSDL 1.1, a `uri` *element information item* or a `serviceQName` *element information item* as defined above.
  - Zero or more namespace-qualified *element information item*s amongst its [children]. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the `FaultDestination` *element information item* is *wsmd:destination*.

3.6 MessageID

The value of the MessageID property must be unique and serves to uniquely identify a message.

The `MessageID` *element information item* has:

- A [local name] of `MessageID`.
- A [namespace name] of "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the `MessageID` *element information item* is *xs:anyURI*.

3.7 MessageReference

MessageReference allows a message to be correlated to another message and specifies the reason for the correlation. The MessageReference property consists of two parts each of which is of type *xs:anyURI*. The first part consists of a MessageID, the ID of the message that triggered the generation of the message containing the MessageReference. The second optional part consists of the Reason for the message and is

of type *xs:anyURI*.

MessageReference is defined by the following pseudo-schema:

```
<wsmd:MessageReference
    wsmd:reason="xs:anyURI"? >xs:anyURI
</wsmd:MessageReference>
```

The `MessageReference` *element information item* has:

- A [local name] of `MessageReference`.
- A [namespace name] of "http://www.w3.org/2004/04/ws-messagedelivery".
- Zero or more *attribute information item*s amongst its [attributes] as follows:
    - An OPTIONAL `reason` *attribute information item*. The [local name] property of which is reason, [namespace name] property is "http://www.w3.org/2004/04/ws-messagedelivery" and the type is *xs:anyURI*.
    - Zero or more namespace qualified *attribute information item*s. The [namespace name] of such *attribute information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the `MessageReference` *element information item* is *wsmd:messageReference*.

There are four predefined values for the `wsmd:reason` attribute:

1. "http://www.w3.org/2004/04/ws-messagedelivery/reason/response" -- implies that the message is a response and the ID of the message in the wsmd:MessageReference identifies the request message.
2. "http://www.w3.org/2004/04/ws-messagedelivery/reason/fault" - implies that the message is a fault and the ID of the message in the wsmd:MessageReference identifies the message that triggered this fault.
3. "http://www.w3.org/2004/04/ws-messagedelivery/reason/notification" -- implies that the message is a notification and the ID of the message in the wsmd:MessageReference identifies the message that requested the notification.
4. "http://www.w3.org/2004/04/ws-messagedelivery/reason/callback" -- implies that the message is a ultimate response message (the Callback message) and the ID of the message in the wsmd:MessageReference identifies the message initial request message.

If omitted, the implicit value for the `reason` attribute is "http://www.w3.org/2004/04/ws-messagedelivery/reason/response".

3.8 OperationName

The value of the OperationName property must be the NCName of the operation specified in WSDL that indicates the specific message exchange. The value of the property in conjunction with the WSRef of the web service that defines the service is used to infer the specific operation uniquely.

The `OperationName` *element information item* has:

- A [local name] of `OperationName`.
- A [namespace name] of "http://www.w3.org/2004/04/ws-messagedelivery".

The type of the `OperationName` *element information item* is *xs:NCName*.

3.9 Mapping of AMDP to SOAP

Each of the six abstract message delivery properties get mapped to a SOAP header block when using SOAP (version 1.1 or 1.2) as the protocol. The processing semantics defined by SOAP 1.1 or SOAP 1.2 apply when AMDP are sent as SOAP headers blocks.

For example:

```
<soap11:Envelope
    xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsdl11="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery">
<soap11:header xmlns:cr="http://example.com/creditbureau">
<wsmd:MessageOriginator>
   <wsmd:uri>http://example.com/app/myMsgOrg</wsmd:uri>
</wsmd:MessageOriginator>

<wsmd:MessageDestination
    wsmd:targetNamespace="http://example.com/creditbureau"
    wsmd:wsdlLocation="http://example.com/creditbureau/wsdl">
   <wsdl11:service name="CreditBureauService"
       wsmd:portType="cr:CreditBureauPT">
       <wsdl11:port binding="cr:SOAPCreditBureauBinding">
          <soapbind:address location="http://example.com/creditbureau/impl"/>
       </wsdl11:port>
   </wsdl11:service>
</wsmd:MessageDestination>

<wsmd:MessageID>http://example.com/myMsgId</wsmd:MessageID>

<wsmd:MessageReference>http://example.com/creditbureau/response-0123456789
</wsmd:MessageReference>

<wsmd:OperationName>generateCreditReport</wsmd:OperationName>
</soap11:header>
```

```
<soap11:body>
...
</soap11:body>
</soap11:Envelope>
```

4 Message Exchange Patterns and AMDP

Message Exchange Patterns describe the message exchanged between nodes, typically Web services and clients of Web services. WSDL describes a message exchange pattern from the perspective of the service, namely the input messages and/or output messages and/or fault messages that can occur in the message exchange.

In order to send a message to a Web service and utilize/implement a message exchange pattern, a client or the service may need to designate the destination of the messages that are specified in a message exchange pattern, such as inputs, outputs and faults. This section defines how AMDPs are used in the context of MEPs defined in WSDL 1.1.

The use of AMDP to implement an MEP MUST NOT violate the binding contract specified for a Web service. A binding contract may specify, although is not limited to, how to target messages towards particular destinations, and send replies/faults in response to a request message. When AMDP is used in conjunction with such a binding, the non-dereferenceable URI value "http://www.w3.org/2004/04/ws-messagedelivery/destination/transport-specified" (defined in **3.1.3 uri element information item** [p.16]) SHOULD be used for appropriate destinations. For example, when using SOAP 1.1 with the SOAP HTTP binding for a request-response operation, the binding specifies that the response message must be sent over the same HTTP connection as the request message. Such an HTTP connection cannot be specified with a dereferenceable URI. In such a case the value for the property ReplyDestination (defined in **3.4 ReplyDestination** [p.19]) SHOULD have the value "http://www.w3.org/2004/04/ws-messagedelivery/destination/transport-specified". This value indicates that the binding rules MUST be used to send the response and the participant is designated in a transport specific manner.

4.1 Declaring the usage of AMDP in WSDL 1.1

In WSDL 1.1 documents, an extension element, `wsmd:MessageDeliveryFeature` is used to indicate that message delivery properties are in effect. This element is defined by using the pseudo-schema:

```
<wsmd:MessageDeliveryFeature/>
```

The extension element can be present as a child element of `operation` (which can be a child element of `wsdl11:portType` or `wsdl11:binding`), or a child element of `wsdl11:binding`. For example,

```
<wsdl11:definitions
    xmlns:wsdl11="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery
    targetNamespace="http://example.com/myNamespace">

    <wsdl11:binding name="myBinding">    . . .
        <wsmd:MessageDeliveryFeature/>
    </wsdl11:binding>
</wsdl11:definitions>
```

The `MessageDeliveryFeature` *element information item* has the following Infoset properties:

- A [local name] of `MessageDeliveryFeature`.
- A [namespace name] of "http://www.w3.org/2004/04/ws-messagedelivery".
- Zero or more namespace qualified *attribute information item*s amongst its [attributes]. The [namespace name] of such *attribute information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".
- Zero or more namespace-qualified *element information item*s amongst its [children]. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".

4.2 WSDL 1.1 Message Exchange Patterns and AMDP

WSDL 1.1 defines a basic set of four Message Exchange Patterns or operations. It is possible to define a *composite MEP* that is composed of predefined operations in WSDL 1.1. Such a composite MEP, when used in conjunction with AMDP, MUST NOT violate the AMDP rules defined for the predefined MEPs. For an example of such a composite MEP see **5 CallBack Pattern** [p.31] .

This section specifies the how AMDP are used in the context of the four operations that are defined in WSDL 1.1. For each message in an operation (’Input’, ’Output’ or ’Fault’), this section defines the mandatory properties, the optional properties and restrictions on the value of the properties. The optional properties MAY be used by a composite transmission primitive to implement the composite transmission primitive.

In every message the value of the property MessageOriginator is used to identify the sender of the message; the value of the property MessageDestination is used by the endpoint implementation to target the recipient of the message and the value of the property MessageID is used to uniquely identify the message.

4.2.1 One-way

One-way transmission primitive/MEP consists of only one *In* message sent by a Node N to the Web service.

The In message MUST have the following properties specified:

- MessageDestination - has the value of the service element that describes the target Web service.
- MessageOriginator - has the value that represents the client or the Web service (Node N) that sent the message.
- OperationName - has the value of the operation that specified the specific message exchange.

The *In* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination
- MessageID
- MessageReference

The optional properties ReplyDestination, FaultDestination, and MessageReference are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

4.2.2 Request-response

Request-response transmission primitive/MEP consists of one *In* message sent by a node N to the Web service and one *Out* message sent by the Web service to Node N. A *fault* message may be sent instead of the *Out* message.

The *In* message MUST have the following properties specified:

- MessageDestination - has the value of the service element that describes the target Web service.
- MessageOriginator - has the value that represents the client or the Web service (node N) that sent the message.
- MessageID
- OperationName - has the value of the operation that specified the specific message exchange.

The *In* message MAY have the following properties specified:

- ReplyDestination - if present, its value MUST be a URI
- FaultDestination - if present, its value MUST be a URI.
- MessageReference

If there is a *fault* generated as a result of processing the message it is sent to the destination identified by FaultDestination, if present, else to the destination identified by MessageOriginator. The response to the *In* message is sent to the destination identified by ReplyDestination, if present, else to the destination identified by MessageOriginator. If any of these two optional properties are not specified, the MessageOriginator property MUST be a URI.

The optional property MessageReference is ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

The *Out/fault* message MUST have the following properties specified:

- MessageDestination

  In case of a *fault* message, the value of this property is the same as that of the FaultDestination property of the In message, if present, else the value of the MessageOriginator property of the *In* message. In case of an *Out* message, the value of this property is the same as that of the ReplyDestination property of the *In* message, if present, else the value of the MessageOriginator property of the *In* message.

- MessageOriginator - has the value of the `service` element that describes the Web service that generated the fault.

- MessageReference - has a value that is the same as the value of the MessageID property of the *In* message. The value of the reason attribute is "http://www.w3.org/2004/04/ws-messagedelivery/reason/fault" for a fault message and "http://www.w3.org/2004/04/ws-messagedelivery/reason/response"

- OperationName - has the value of the operation that specified the specific message exchange.

The *Out/fault* message MAY have the following properties specified:

- ReplyDestination

- FaultDestination

- MessageID

The optional properties ReplyDestination, FaultDestination, and MessageID are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

4.2.3 Notification

Notification transmission primitive/MEP consists of only one *Out* message sent by the Web service to Node N.

The *Out* message MUST have the following properties specified:

- MessageDestination - has the value that represents the client or the Web service that will receive the message (Node N)

- MessageOriginator - has the value of the `service` element that describes the Web service which is the sender of the message.

- OperationName - has the value of the operation that specified the specific message exchange.

The *Out* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination
- MessageID
- MessageReference

The optional properties ReplyDestination, FaultDestination, MessageID, and MessageReference are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

4.2.4 Solicit-response

Solicit-response transmission primitive/MEP consists of one *Out* message sent by the Web service to Node N and one *In* message sent by Node N back to the Web service. A *fault* message may be sent instead of the *In* message.

The *Out* message MUST have the following properties specified:

- MessageDestination - has the value that represents the client or the Web service that will receive the message (Node N)
- MessageOriginator - has the value of the `service` element that describes the Web service which is the sender of the message.
- MessageID
- OperationName - has the value of the operation that specified the specific message exchange.

The *Out* message MAY have the following properties specified:

- ReplyDestination - If present, its value MUST be a URI
- FaultDestination - If present, its value MUST be a URI.
- MessageReference

The optional property MessageReference is ignored when implementing this MEP, but MAY be used when implementing a composite MEP. If there is a *fault* generated as a result of processing the message it is sent to the destination identified by FaultDestination, if present, else to the destination identified by MessageOriginator. The response to the *Out* message is sent to the destination identified by ReplyDestination, if present, else to the destination identified by MessageOriginator.

The *In*/*fault* message MUST have the following properties specified:

- MessageDestination

In case of a *fault* message, the value of this property is the same as that of the FaultDestination property of the *Out* message, if present, else the value of the MessageOriginator property of the *Out* message. In case of an *In* message, the value of this property is the same as that of the ReplyDestination property, of the *Out* message, if present, else the value of the MessageOriginator property of the *Out* message.

- MessageOriginator - has the same value as that of the MessageDestination property of the *Out* message
- MessageReference - has a value that is the same as the value of the MessageID property of the *Out* message. The value of the reason attribute is "http://www.w3.org/2004/04/ws-messagedelivery/reason/fault" for the *fault* message and "http://www.w3.org/2004/04/ws-messagedelivery/reason/response" for an *In* message.
- OperationName - has the value of the operation that specified the specific message exchange.

The *In*/*fault* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination
- MessageID

The optional properties ReplyDestination, FaultDestination, and MessageID are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

4.2.5 Summary

The table below summarizes the properties used in the context of WSDL 1.1 operations and the restrictions on the value of those properties.

AMDP for WSDL 1.1 MEPs

		MO	MD	RD	FD	MID	MR	ON
One-way		R	R (WSRef)	------	------	------	------	R
Request-Response	**Input**	R+	R (WSRef)	O (URI)	O (URI)	R	------	R
	Output	R (WSRef)	R	------	------	------	R	R
	Fault	R (WSRef)	R	------	------	------	R	R
Notification		R (WSRef)	R	------	------	------	------	R
Solicit-Response	**Output**	R (WSRef)	R	O (URI)	O (URI)	R	------	R
	Input	R	R (WSRef)	------	------	------	R	R
	Fault	R	R (WSRef)	------	------	------	R	R

AMDP Acronyms:

MO = MessageOriginator; MD = MessageDestination; RD = ReplyDestination; FD = FaultDestination; MID = MessageID; MR = MessageReference; ON = OperationName

Value Designators:

R = Required; O = Optional; -- = Ignored; (...) = Type of the value; WSRef = WSDL 1.1 service element or QName; + = must be a URI if any of the optional non-ignored properties is absent

Red color = required value; Blue color = optional value;

5 CallBack Pattern

A Callback pattern is used to asynchronously deliver a response message to a request message. This is a very useful and prevalent scenario, especially when the delay between the time that a request is received and a response is generated and sent to the initial requestor, is large or non-deterministic.

This pattern consist of an *initial* request message that is sent by the initial requestor to a responding Web service followed by an *ultimate* response message that is sent by the responding Web service to the requesting Web service (which is the same as the initial requestor). The ultimate response message is the final response that is sent to destination that needs to receive the response. The initial request message and the ultimate response message may use different transports/bindings.

Note:

Although the request-response pattern in WSDL 1.1 may be used in conjunction with an asynchronous transport and hence binding, it is impossible to ensure the usage of two different bindings (such as SOAP/HTTP and SMTP) by utilizing the request-response pattern directly to implement a callback pattern. When different bindings are necessary for the two message exchanges that would constitute a callback pattern, the techniques illustrated here are necessary.

In order to implement such a pattern, in most cases, the initial requestor has to include in the initial request message:

1. A reference to the endpoint/Web service, called as the ultimate response destination, to which the ultimate response message is to be sent.

2. Information that can be included in the ultimate response message, *called the correlation id*, which can be used by the ultimate response destination to correlate the ultimate response message with the initial request message.

An application can implement a Callback pattern in three different ways:

1. A URI/WSRef that points to the ultimate response destination is embedded in the initial request message in an application specific way along with an application specific correlation id. The Web service that receives the initial request message then has to implement the Callback pattern in an application specific way.

2. It may use the request-response operation in WSDL 1.1 and use a binding/transport specific way to specify the correlation id and ultimate response destination in the initial request message.

3. It may use an application, transport and binding independent way to implement the Callback pattern.

The first approach, although important to specific applications, is not used, because it generates ad hoc solutions and it is difficult to automate at the infrastructure level. As a result, this approach inhibits interoperability. Therefore, it is necessary to develop an application independent mechanism for Callbacks. The second approach requires transport/binding specific mechanisms and cannot use different transports/bindings for sending the initial request message and the ultimate response message. The third is the most general as it does not rely on a specific layering on transport properties. Our definition of

Callback pattern uses the third approach by incorporating a WSDL centric solution that utilizes AMDPs for implementing Callbacks. This approach decouples the application logic from the implementation of Callbacks, is transport layering independent and enables tooling for this pattern.

The approach taken is an example of utilizing AMDP to implement composite MEPs. The Callback pattern consists of two Web service portTypes (operations) that are tied together to represent the Callback pattern (or composite MEP) -- the responding Web service that receives and processes the initial request message (and generates the response) and the requesting Web service (which is the ultimate message destination) that receives and processes the ultimate response message (and that sent the initial request message).

A WSDL extensibility element `wsmd:ResponseOperation` is defined to specify the Callback relationship between the two operations in WSDL 1.1.

The Callback pattern is implemented in two different ways -

1. correlating two one-way operations in the case of WSDL 1.1

2. correlating two request-response operations in the case of WSDL 1.1

5.1 Declaring Callbacks in WSDL Documents

In WSDL documents, an extension element, `wsmd:ResponseOperation` is used to indicate that the Callback pattern is being used. This extensibility element specifies the portType and the operation (and optionally binding) used for sending the ultimate response message.

For WSDL 1.1, the extensibility element can be present as a child element of `wsdl11:operation` (where the `wsdl11:operation` is either a child element of `wsdl11:portType` or `wsdl11:binding`) that describe the responding Web service.

For examples of use of this extensibility element see sections **5.2 Callback Representation Using Two Operations with Input Messages** [p.34] and **5.3 Callback Representation Using Two Operations with Input-Output Messages** [p.35] .

5.1.1 `wsmd:ResponseOperation` Extensibility Element

The `wsmd:ResponseOperation` extensibility element is defined by the following pseudo-schema:

```
<wsmd:ResponseOperation
    wsmd:interface="xs:QName"
    wsmd:operation="xs:NCName"
    wsmd:binding="xs:QName"?/>
```

The ResponseOperation *element information item* has the following Infoset properties:

- A [local name] of `ResponseOperation`.

- A [namespace name] of "http://www.w3.org/2004/04/ws-messagedelivery".
- Two or more *attribute information item*s amongst its [attributes] as follows:
  - A MANDATORY `interface` *attribute information item*. The [local name] property of which is `interface`, the [namespace name] property is "http://www.w3.org/2004/04/ws-messagedelivery", and the type is *xs:QName*.
  - A MANDATORY `operation` *attribute information item*. The [local name] property of which is `operation`, [namespace name] property is "http://www.w3.org/2004/04/ws-messagedelivery", and the type is *xs:NCName*.
  - An OPTIONAL `binding` *attribute information item*. The [local name] property of which is `binding`, [namespace name] property is "http://www.w3.org/2004/04/ws-messagedelivery", and the type is *xs:QName*.
  - Zero or more namespace qualified *attribute information item*s. The [namespace name] of such *attribute information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".
- Zero or more namespace-qualified *element information item*s amongst its [children]. The [namespace name] of such *element information item*s MUST NOT be "http://www.w3.org/2004/04/ws-messagedelivery".

The value of the mandatory *attribute information item* `wsmd:interface` is a QName which identifies the portType of the requesting Web service, in the case of WSDL 1.1.

The value of the mandatory *attribute information item* `wsmd:operation` is a NCName which identifies the `operation` of the requesting Web service (within the interface/portType identified by the `wsmd:interface` attribute).

The value of the optional *attribute information item* `wsmd:binding` is a QName which identifies the `binding` of the requesting Web service.

The `wsmd:ResponseOperation` element may appear in the following three locations in a WSDL document, within an `operation` in a `portType` that defines the request, an `operation` within a `binding` or a `port` within a `service`. The second option may used to specify the binding of the requesting Web service. This separation is for relating the two operations the abstract layer within WSDL independently of the binding. The third option may be used to indicate the portType and the binding of the response at the `service` element of the defining service. The `wsmd:binding` attribute MUST NOT be present when `wsmd:ResponseOperation` element appears as an extensibility element in an `operation` within a `portType`. The `wsmd:ResponseOperation` attribute MUST NOT contradict each other when it is specified within multiple locations in WSDL, such as a `portType` and its related `binding`.

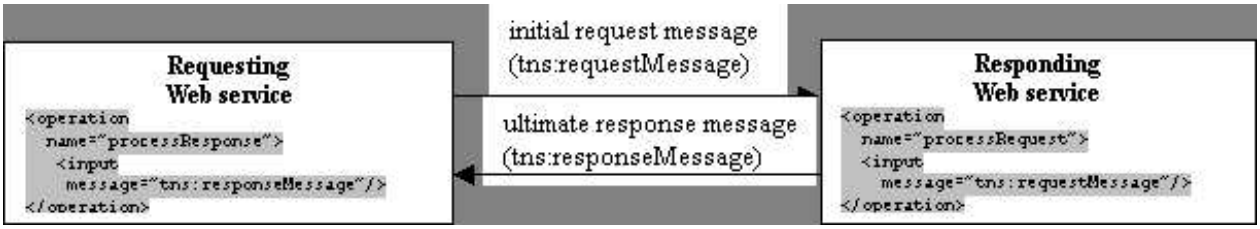
5.2 Callback Representation Using Two Operations with Input Messages

The example below illustrates the use of two operations with input messages to represent Callback.

```
<wsdl11:definitions
    xmlns:wsdl11="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery"
    xmlns:tns="http://example.com/callback/2-input"
    targetNamespace="http://example.com/callback/2-input">
. . .
<wsdl11:portType name="processRequestPortType">
    <wsdl11:operation name="processRequest">
         <wsdl11:input message="tns:requestMessage"/>         <wsmd:ResponseOperation interface="tns:processResponsePortType"
                                operation="processResponse"/>
         <wsmd:MessageDeliveryFeature/>
    </wsdl11:operation>
</wsdl11:portType>

<wsdl11:portType name="processResponsePortType">
     <wsdl11:operation name="processResponse">
          <wsdl11:input message="tns:responseMessage"/>...          <wsmd:MessageDeliveryFeature/>
      </wsdl11:operation>
</wsdl11:portType>

<wsdl11:service name="CBService">
    <wsdl11:port name="CBPort">
        <soap11:address location="http://example.com/cb-impl1"/>
    </wsdl11:port>
</wsdl11:service>
</wsdl11:definitions>
```

In this example, there are two portTypes. The portType `processRequestPortType` (implemented by the responding Web service) contains the operation `processRequest` that is correlated to the operation `processResponse` in portType `processResponsePortType` (implemented by the requesting Web service). The extensibility element `wsmd:ResponseOperation` specifies the portType and the operation that is used by the requesting Web service. Web service `CBService` implements the operation `processRequest`. When this service is invoked, the AMDP SOAP headers provide the necessary information about the correlation id that is to be used in the ultimate response message and the destination (which will include the binding and endpoint information) where the ultimate response must be sent.

The figure below depicts the messages sent between the requesting Web service and the responding Web service when the Callback pattern is implemented using two operations with input messages.



To represent Callback using two input messages:

- The extensibility element `wsmd:ResponseOperation` MUST be present in the wsdl11:operation element. This operation is implemented by the responding Web service.

- The responding Web service and requesting Web service MUST implement the AMDP feature.

The initial request message sent to the responding Web service MUST have the value of the `wsmd:MessageOriginator` property set to represent the requesting Web service. The responding Web service uses the interface/operation/binding information provided by the `wsmd:ResponseOperation` extensibility element as well as the value of the `wsmd:MessageOriginator` property (which is only available at runtime) to invoke the requesting Web service and send the ultimate response message. It is an error if the value of the `wsmd:MessageOriginator` property of the initial request message does not support/implement the portType/operation/binding specified by the `wsmd:ResponseOperation` extensibility element in the WSDL description.

The initial request message MUST specify a value for the `wsmd:MessageID` property. This value is used as the correlation id and appears as the value of the `wsmd:MessageReference` property of the ultimate response message. The `wsmd:reason` attribute value of the ultimate response message is "http://www.w3.org/2004/04/ws-messagedelivery/reason/callback".

5.3 Callback Representation Using Two Operations with Input-Output Messages

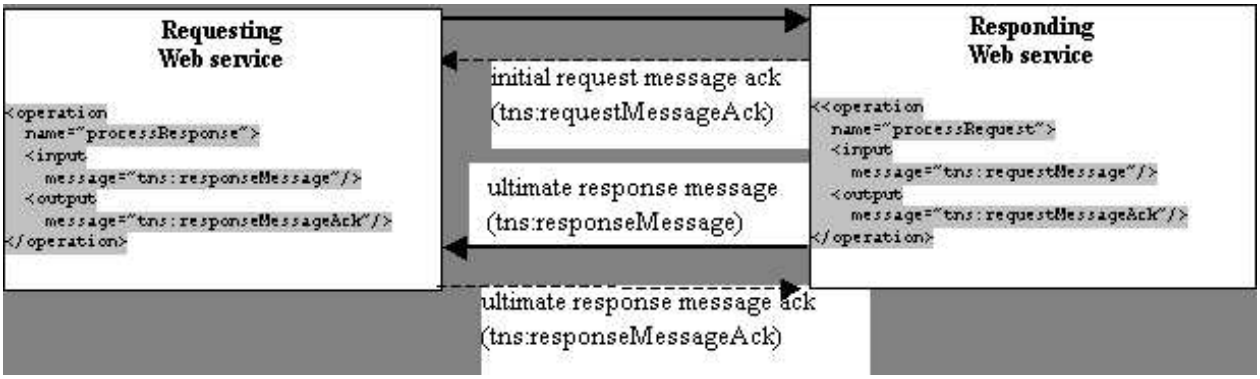
The example below illustrates the use of two input-output operations to represent Callback.

```
<wsdl11:definitions
    xmlns:wsdl11="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery"
    xmlns:tns="http://example.com/callback/2-input-output"
    targetNamespace="http://example.com/callback/2-input-output">
. . .
<wsdl11:portType name="processRequestPortType">
    <wsdl11:operation name="processRequest">
        <wsdl11:input message="tns:requestMessage"/>
        <wsdl11:output message="tns:requestMessageAck"/>        <wsmd:ResponseOperation interface="tns:processResponsePortType"
                                operation="processResponse"/>
        <wsmd:MessageDeliveryFeature/>
    </wsdl11:operation>
</wsdl11:portType>

<wsdl11:portType name="processResponsePortType">
    <wsdl11:operation name="processResponse">
        <wsdl11:input message="tns:responseMessage"/>
        <wsdl11:output message="tns:responseMessageAck"/>        <wsmd:MessageDeliveryFeature/>
    </wsdl11:operation>
</wsdl11:portType>

<wsdl11:service name="CBService">
    <wsdl11:port name="CBPort">
        <soap11:address location="http://example.com/cb-impl2"/>
    </wsdl11:port>
</wsdl11:service>
</wsdl11:definitions>
```

In this example, there are two portTypes. portType `processRequestPortType` (implemented by the responding Web service) contains the operation `processRequest` which is correlated to the operation `processResponse` in portType `processResponsePortType` (implemented by the requesting Web service). The extensibility element `wsmd:ResponseOperation` specifies the portType and the operation that is used by the requesting Web service. Web service `CBService` implements the operation `processRequest`. When this service is invoked, the AMDP SOAP headers provide the necessary information about the correlation id that is to be used in the ultimate response message and the destination

(which will include the binding and endpoint information) where the ultimate response must be sent.

The figure below depicts the messages sent between the requesting Web service and the responding Web service when the Callback pattern is implemented using two operations with input messages.



To represent Callback using a pair of input-output messages:

- The extensibility element `wsmd:ResponseOperation` MUST be present in the `wsdl11:operation` element. This operation is implemented by the responding Web service.
- The responding Web service and requesting Web service MUST implement the AMDP feature.

The initial request message sent to the responding Web service MUST have the value of the `wsmd:MessageOriginator` property set to represent the requesting Web service. The responding Web service immediately sends a message back to the requesting Web service using the same binding -- this message is an acknowledgement of the receipt of the initial request and *is not the Callback message*. The responding Web service uses the portType/operation/binding information provided by the `wsmd:ResponseOperation` extensibility element as well as the value of the `wsmd:MessageOriginator` property (which is only available at runtime) to invoke the requesting Web service and send the ultimate response message. It is an error if the value of the `wsmd:MessageOriginator` property of the initial request message does not support/implement the interface/operation/binding specified by the `wsmd:ResponseOperation` extensibility element in the WSDL description. The requesting Web service then sends an acknowledgement of the Callback message to the responding Web service.

The initial request message MUST specify a value for the `wsmd:MessageID` property. This value is used as the correlation id and appears as the value of the `wsmd:MessageReference` property of the ultimate response message. The `wsmd:reason` attribute value of the ultimate response message is "http://www.w3.org/2004/04/ws-messagedelivery/reason/callback".

The initial request message MUST specify a value for the `wsmd:FaultDestination` property and the `wsmd:ReplyDestination` property. The `wsmd:MessageOriginator` property value of the initial request message identifies the ultimate response destination and not the fault or reply destination.

6 Message Delivery and WSDL 2.0

Note:

At the time this document is written, WSDL 2.0 specification is still in progress. Hence, the discussion in this section and details that are presented in Appendix B are subject to change and may not apply if the MEP definitions change in WSDL 2.0.

WSDL 2.0 [WSDL 2.0 Part 2] [p.40] formally defines a basic set of seven MEPs. It also allows defining new MEPs. WSDL 2.0 uses the following namespace-prefix in addition to Table 1:

Additional Namespace-Prefix Bindings for WSDL 2.0

Prefix	Namespace
wsdl20	"http://www.w3.org/2004/03/wsdl"

The ideas presented in the previous Sections apply in the same manner to WSDL 2.0 MEPs with the following changes:

- **WSRefs:** The WSRef in WSDL 2.0 naturally depend on the definition of the service element/type defined in the WSDL 2.0 schema as specified by [WSDL 2.0 Part 1] [p.40] and it is different than service references defined here for WSDL 1.1 due to schema differences. Whenever WSRefs are utilized with AMDPs, the corresponding WSRef definition for WSDL 2.0 must be used with services that use WSDL 2.0 descriptions. Please refer to **B Appendix: WSDL 2.0 and AMDPs** [p.44] for more discussion on service References with WSDL 2.0.

- **AMDP content changes:** Currently, all AMDPs that are based on *wsmd:destination* type is defined by using `wsdl1:service` element. In addition to `wsdl1:service` element, a *wsmd:destination* may be a WSDL 2.0 WSRef defined by `wsdl20:service` element and may appear wherever `wsdl11:service` element is allowed. These properties based on *wsmd:destination* are:

  - `wsmd:MessageOriginator`
  - `wsmd:MessageDestination`
  - `wsmd:ReplyDestination`
  - `wsmd:FaultDestination`

- **MEPs in WSDL 2.0:** There are 7 different MEPs in the current draft of WSDL 2.0 specification. The changes that are discussed above can be applied to all the patterns and AMDPs may be utilized for all WSDL 2.0 patterns. The specifics of how AMDPs may be used in conjunction to the WSDL 2.0 patterns are discussed in Appendix B in detail.

- **Definition of Callback Pattern:** The definition of callback pattern as specified in **5 CallBack Pattern** [p.31] apply when WSDL 2.0 based Web services participate in a MEP. All constraints of the Callback pattern and its relationship to AMDP defined in **5 CallBack Pattern** [p.31] apply using the In (instead of One-way in WSDL 1.1) and In-Out pattern (instead of Request-Response in WSDL 1.1) patterns. Note that it is possible to use an In-Out pattern in conjunction with asynchronous binding/transport instead of the Callback composite pattern (as mentioned in **5 CallBack Pattern** [p.31] in the context of WSDL 1.1). However, such an implementation has a restriction that the *In* and *Out* message *must use the same binding*. This restriction does not exist in the case of the Callback composite pattern.

  The values of the attributes of `wsmd:ResponseOperation` must be as follows:

  - `wsmd:interface`: QName of the interface of the requesting Web service defined by WSDL 2.0 that defines the where the response operation exists.
  - `wsmd:operation`: NCName of the operation of the requesting Web service defined by the interface indicated by the `wsmd:interface` attribute that specifies the response operation.
  - `wsmd:binding`: QName of the binding of the requesting Web service.

AMDP usage in WSDL 2.0 may be indicated by using the Features and Properties syntax. The AMDPs that are specified in this document may be regarded as Properties that the WS-MessageDelivery feature specifies and provides.

7 Security Considerations

In addition to the security consideration that have to be taken into account for Web services in general, additional attention has to be paid to the AMDP expressed in the messages. AMDP in a message can specify the destination for the message, reply and/or faults that may be generated when the message is processed. A message may pass through multiple intermediaries. A malicious intermediary may modify the AMDP associated with the message. It is therefore necessary to take the AMDP headers into account when mechanisms that ensure integrity, privacy and authenticity of the message are considered. For example, when digitally signing a message, attention should be paid to the AMDP to determine if the AMDP must be signed along with the rest of the message. AMDP headers can be abused to spoof destinations. Situations where this is possible SHOULD resort to digitally signing the messages and ensuring that the signer has the authority to set the destinations in the AMDP headers.

8 References

WSDL 2.0 Part 1

Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, R. Chinnici, M. Gudgin, J-J. Moreau, J. Schlimmer, and S. Weerawarana, Editors. World Wide Web Consortium, 26 March 2004. This version of the "Web Services Description Version 2.0: Core Language" Specification is http://www.w3.org/TR/2004/WD-wsdl20-20040326. The latest version of "Web Services Description Version 2.0: Core Language" is available at http://www.w3.org/TR/wsdl20.

WSDL 2.0 Part 2

Web Services Description Language (WSDL) Version 2.0 Part 2: Message Exchange Patterns, M. Gudgin, A. Lewis, and J. Schlimmer, Editors. World Wide Web Consortium, 26 March 2004. This version of the "Web Services Description Version 2.0: Message Exchange Patterns" Specification is available at http://www.w3.org/TR/2004/WD-wsdl20-patterns-20040326. The latest version of "Web Services Description Version 2.0: Message Exchange Patterns" is available at http://www.w3.org/TR/wsdl20-patterns.

WSDL 1.1

Web Services Description Language (WSDL) 1.1, E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, Authors. World Wide Web Consortium, 15 March 2001. This version of the Web Services Description Language 1.1 Note is http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

SOAP 1.1

Simple Object Access Protocol (SOAP) 1.1, D. Box, et al., Authors. World Wide Web Consortium, 8 May 2000. This version of the Simple Object Access Protocol (SOAP) 1.1 Specification is http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

XML Schema: Structures

XML Schema Part 1: Structures, H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, Editors. World Wide Web Consortium, 2 May 2001. This version of the XML Schema Part 1 Recommendation is http://www.w3.org/TR/2001/REC-xmlschema-1-20010502. The latest version of XML Schema Part 1 is available at http://www.w3.org/TR/xmlschema-1.

XML Schema: Datatypes

XML Schema Part 2: Datatypes, P. Byron, and A. Malhotra, Editors. World Wide Web Consortium, 2 May 2001. This version of the XML Schema Part 2 Recommendation is http://www.w3.org/TR/2001/REC-xmlschema-2-20010502. The latest version of XML Schema Part 2 is available at http://www.w3.org/TR/xmlschema-2.

BP 1.0

Basic Profile Version 1.0a, K. Ballinger, D. Ehnebuske, M. Gudgin, M. Nottingham, and P. Yendluri, Editors. The Web Services-Interoperability Organization, 8 August 2003. This version of the Basic Profile Version 1.0a Final Specification is http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.html.

IETF RFC 2119

Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, Author. Internet Engineering Task Force, June 1999. Available at http://www.ietf.org/rfc/rfc2119.txt.

XML Information Set

XML Information Set, J. Cowan, and R. Tobin, Editors. World Wide Web Consortium, 24 October 2001. This version of the XML Information Set Recommendation is http://www.w3.org/TR/2001/REC-xml-infoset-20011024. The latest version of XML Information Set is available at http://www.w3.org/TR/xml-infoset.

9 Acknowledgements

The authors would like to thank Doug Bunting, Martin Chapman, Roberto Chinnici, Jacques Durand, Marc Hadley, Bill Jones, Sunil Kunisetty, Sastry Malladi, Greg Pavlik, Ekkehard Rohwedder, and Tom Rutt for their constructive comments and review.

A Appendix: Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:wsmd="http://www.w3.org/2004/04/ws-messagedelivery"
    xmlns:wsdl11="http://schema.xmlsoap.org/wsdl/"
    targetNamespace="http://www.w3.org/2004/04/ws-messagedelivery"
    elementFormDefault="qualified">

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

<!-- AMDP SOAP header blocks -->
<xs:element name="MessageOriginator" type="wsmd:destination"/>
<xs:element name="MessageDestination" type="wsmd:destination"/>
<xs:element name="ReplyDestination" type="wsmd:destination"/>
<xs:element name="FaultDestination" type="wsmd:destination"/>
<xs:element name="MessageID" type="xs:anyURI"/>
<xs:element name="MessageReference" type="wsmd:messageReference"/>
<xs:element name="OperationName" type="wsmd:operationName"/>

<!-- Top level types -->
<xs:complexType name="destination">
    <xs:sequence>
        <xs:choice minOccurs="1" maxOccurs="1">
            <xs:element ref="wsdl11:service"/>
            <xs:element name="uri" type="xs:anyURI"/>
            <xs:element name="serviceQName" type="xs:QName"/>
        </xs:choice>
        <xs:any namespace="##other" minOccurs="0"
                maxOccurs="unbounded" processContents="lax"/>
    </xs:sequence>
    <xs:attribute ref="wsmd:targetNamespace"/>
    <xs:attribute ref="wsmd:wsdlLocation"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<xs:complexType name="messageReference">
    <xs:simpleContent>
        <xs:extension base="xs:anyURI">
           <xs:attribute name="reason" type="xs:anyURI" form="qualified"/>
           <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:complexType name="operationName">
    <xs:simpleContent>
        <xs:extension base="xs:NCName">
           <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<!-- WSDL extensibility elements -->
<xs:element name="MessageDeliveryFeature">
    <xs:complexType>
```

```
        <xs:sequence>
            <xs:any namespace="##other" minOccurs="0"
                    maxOccurs="unbounded" processContents="lax"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other"
                    processContents="lax"/>
    </xs:complexType>
</xs:element>

<xs:element name="ResponseOperation">
     <xs:complexType>
         <xs:sequence>
             <xs:any namespace="##other" minOccurs="0"
                     maxOccurs="unbounded" processContents="lax"/>
         </xs:sequence>
         <xs:attribute name="interface" type="xs:QName" use="required"/>
         <xs:attribute name="operation" type="xs:NCName" use="required"/>
         <xs:anyAttribute namespace="##other" processContents="lax"/>
     </xs:complexType>
</xs:element>

<xs:element name="portType">
     <xs:complexType>
          <xs:simpleContent>
              <xs:extension base="xs:QName">
                  <xs:anyAttribute namespace="##other" processContents="lax"/>
              </xs:extension>
          </xs:simpleContent>
     </xs:complexType>
</xs:element>

<!-- top level attributes -->
<xs:attribute name="targetNamespace" type="xs:anyURI"/>
<xs:attribute name="wsdlLocation">
    <xs:simpleType>
       <xs:list itemType="xs:anyURI"/>
    </xs:simpleType>
</xs:attribute>
</xs:schema>
```

B Appendix: WSDL 2.0 and AMDPs

B.1 WSRefs in WSDL 2.0

Note:

WSRefs and their representations have been accepted by the WSD WG, but are not yet included in WSDL 2.0 WG drafts. It is expected that they will be in the future WG drafts.

The `service` element in WSDL 2.0 is defined by using *wsdl20:serviceType* in WSDL 2.0 schema. A WSRef in the context of WSDL 2.0 will be represented using the *wsdl20:serviceType* defined by WSDL 2.0. Using XML Schema derivation by restriction, it is possible to lock the interface and/or a specific binding in a reference declaration. It is expected that the WSDL 2.0 specification will illustrate the derivation of specific interface and/or bindings that define a specific Web service reference in WSDL 2.0.

With services that use WSDL 2.0 descriptions, a WSRef that appears as a destination may use `wsdl20:service` element as the content of *wsmd:destination*. In contrast to WSDL 1.1 WSRefs, the Schema definition of a `wsdl20:service` element is restricted to a single interface and a binding is required to cover all the operations in an interface, in contrast to the WSDL 1.1 definition of `wsdl1:service`. Hence the WSRefs in WSDL 2.0 comply with the requirements stated for a WSRef in **2.1 WSRefs in WSDL 1.1** [p.11] without additional requirements.

B.2 WSDL 2.0 MEPs and AMDP

Note:

This section illustrates how AMDP can be used within the context of WSDL 2.0 and shows its applicability of this specification in the future, however it is subject to change due to current status of WSDL 2.0. The WSDL 2.0 specification, the number of MEPs it defines and its schema that this document use as basis may be changed since the WSDL 2.0 specification is not yet final. Therefore, the reader is encouraged to take the discussion as an illustration of the applicability of the technology presented, but not as a final requirement for using AMDPs with WSDL 2.0 descriptions.

WSDL 2.0 defines a basic set of seven MEPs. It also allows defining new MEPs. A WSDL 2.0 MEP is specified on the interface operation component using the ’pattern’ attribute.

This section specifies how AMDP are used in the context of the seven MEPs that are defined in WSDL 2.0. For each message in an MEP (’In’, ’Out’ or ’Fault’), this section defines the mandatory properties, optional properties, restrictions on the value of the properties, and properties whose values are ignored. The optional/ignored properties MAY be used by a composite MEP to implement the composite MEP. In every message the value of the property MessageOriginator is used to identify the sender of the message; the value of the property MessageDestination is used by the endpoint implementation to target the recipient of the message and the value of the property MessageID is used to uniquely identify the message.

B.2.1 In-Only Pattern

In-Only pattern consists of only one *In* message sent by a Node N to the Web service.

The *In* message MUST have the following properties specified:

- MessageDestination - has the value of the service element that describes the target Web service.
- MessageOriginator - has the value that represents the client or the Web service (Node N) that sent the message.
- OperationName - has the value of the operation that specified the specific message exchange.

The *In* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination
- MessageID
- MessageReference

The optional properties ReplyDestination, FaultDestination, and MessageReference are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

B.2.2 Robust In-Only Pattern

Robust In-only consists of one *In* message sent by a Node N to the Web service and possibly one *fault* message sent in the opposite direction.

The *In* message MUST have the following properties specified:

- MessageDestination - has the value of the service element that describes the target Web service.
- MessageOriginator - has the value that represents the client or the Web service (Node N) that sent the message.
- MessageID
- OperationName - has the value of the operation that specified the specific message exchange.

The *In* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination - if present, its value MUST be a URI.

- MessageReference

If there is a *fault* generated as a result of processing the message it is sent to the destination identified by FaultDestination, if present, else to the destination identified by MessageOriginator. If FaultDestination is not specified, the MessageOriginator property MUST be a URI.

The optional properties ReplyDestination, and MessageReference are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

The *fault* message MUST have the following properties specified:

- MessageDestination - has the same value as that of the FaultDestination property of the *In* message, if present, else the value of the MessageOriginator property of the *In* message.
- MessageOriginator - has the value of the service element that describes the Web service that generated the *fault*.
- MessageReference - has a value that is the same as the value of the MessageID property of the *In* message. The value of the reason attribute is "http://www.w3.org/2004/04/ws-messagedelivery/reason/fault"
- OperationName - has the value of the operation that specified the specific message exchange.

The *fault* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination
- MessageID

The optional properties ReplyDestination, FaultDestination, and MessageID are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

B.2.3 In-out Pattern

In-Out pattern consists of one *In* message sent by a Node N to the Web service and one *Out* message sent by the Web service to Node N. A *fault* message may be sent instead of the *Out* message.

The *In* message MUST have the following properties specified:

- MessageDestination - has the value of the service element that describes the target Web service.
- MessageOriginator - has the value that represents the client or the Web service (Node N) that sent the message.
- MessageID

- OperationName - has the value of the operation that specified the specific message exchange.

The *In* message MAY have the following properties specified:

- ReplyDestination - if present, its value MUST be a URI
- FaultDestination - if present, its value MUST be a URI.
- MessageReference

If there is a *fault* generated as a result of processing the message it is sent to the destination identified by FaultDestination, if present, else to the destination identified by MessageOriginator. The response to the *In* message is sent to the destination identified by ReplyDestination, if present, else to the destination identified by MessageOriginator. If any of these two optional properties are not specified, the MessageOriginator property MUST be a URI.

The optional property MessageReference is ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

The *Out*/*fault* message MUST have the following properties specified:

- MessageDestination

  In case of a *fault* message, the value of this property is the same as that of the FaultDestination property of the *In* message, if present, else the value of the MessageOriginator property of the *In* message. In case of an *Out* message, the value of this property is the same as that of the ReplyDestination property of the *In* message, if present, else the value of the MessageOriginator property of the *In* message.

- MessageOriginator - has the value of the service element that describes the Web service that generated the *fault*.
- MessageReference - has a value that is the same as the value of the MessageID property of the *In* message. The value of the reason attribute is "http://www.w3.org/2004/04/ws-messagedelivery/reason/fault" for a *fault* message and "http://www.w3.org/2004/04/ws-messagedelivery/reason/response"
- OperationName - has the value of the operation that specified the specific message exchange.

The *Out*/*fault* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination
- MessageID

The optional properties ReplyDestination, FaultDestination, and MessageID are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

B.2.4 Out-Only

Out-Only pattern consists of only one *Out* message sent by the Web service to Node N.

The *Out* message MUST have the following properties specified:

- MessageDestination - has the value that represents the client or the Web service that will receive the message (Node N)
- MessageOriginator - has the value of the `service` element that describes the Web service which is the sender of the message.
- OperationName - has the value of the operation that specified the specific message exchange.

The *Out* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination
- MessageID
- MessageReference

The optional properties ReplyDestination, FaultDestination, MessageID, and MessageReference are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

B.2.5 Robust Out-Only

Robust Out-only consists of one *Out* message sent by the Web service to Node N and possibly one *fault* message sent in the opposite direction.

The *Out* message MUST have the following properties specified:

- MessageDestination - has the value that represents the client or the Web service that will receive the message (Node N)
- MessageOriginator - has the value of the service element that describes the Web service which is the sender of the message.
- MessageID
- OperationName - has the value of the operation that specified the specific message exchange.

The *Out* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination - if present, its value MUST be a URI.
- MessageReference

The optional properties ReplyDestination, and MessageReference are ignored when implementing this MEP, but MAY be used when implementing a composite MEP. If there is a *fault* generated as a result of processing the message it is sent to the destination identified by FaultDestination, if present, else to the destination identified by MessageOriginator.

The *fault* message MUST have the following properties specified:

- MessageDestination - has the same value as that of the FaultDestination property, if present, of the *In* message, else the value of the MessageOriginator property of the *In* message.
- MessageOriginator - has the same value as that of the MessageDestination property of the *Out* message if present
- MessageReference - has a value that is the same as the value of the MessageID property of the *Out* message. The value of the reason attribute is "http://www.w3.org/2004/04/ws-messagedelivery/reason/fault"
- OperationName - has the value of the operation that specified the specific message exchange.

The *fault* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination
- MessageID

The optional properties ReplyDestination, FaultDestination, and MessageID are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

B.2.6 Out-In

Out-In pattern consists of one *Out* message sent by the Web service to Node N and one *In* message sent by Node N back to the Web service. A *fault* message may be sent instead of the *In* message.

The *Out* message MUST have the following properties specified:

- MessageDestination - has the value that represents the client or the Web service that will receive the message (Node N)

- MessageOriginator - has the value of the `service` element that describes the Web service which is the sender of the message.
- MessageID
- OperationName - has the value of the operation that specified the specific message exchange.

The *Out* message MAY have the following properties specified:

- ReplyDestination - if present, its value must be a URI
- FaultDestination - if present, its value MUST be a URI.
- MessageReference

The optional property MessageReference is ignored when implementing this MEP, but MAY be used when implementing a composite MEP. If there is a *fault* generated as a result of processing the message it is sent to the destination identified by FaultDestination, if present, else to the destination identified by MessageOriginator. The response to the *Out* message is sent to the destination identified by ReplyDestination, if present, else to the destination identified by MessageOriginator.

The *In/fault* message MUST have the following properties specified:

- MessageDestination

  In case of a *fault* message, the value of this property is the same as that of the FaultDestination property of the *Out* message, if present, else the value of the MessageOriginator property of the *Out* message. In case of an *In* message, the value of this property is the same as that of the ReplyDestination property, of the *Out* message, if present, else the value of the MessageOriginator property of the *Out* message.

- MessageOriginator - has the same value as that of the MessageDestination property of the *Out* message
- MessageReference - has a value that is the same as the value of the MessageID property of the *Out* message. The value of the reason attribute is "http://www.w3.org/2004/04/ws-messagedelivery/reason/fault" for the *fault* message and "http://www.w3.org/2004/04/ws-messagedelivery/reason/response" for an *In* message.
- OperationName - has the value of the operation that specified the specific message exchange.

The *In/fault* message MAY have the following properties specified:

- ReplyDestination
- FaultDestination

- MessageID

The optional properties ReplyDestination, FaultDestination, and MessageID are ignored when implementing this MEP, but MAY be used when implementing a composite MEP.

B.2.7 Out-Optional-In

The Out-Optional-In pattern consists of an *Out* message sent by the Web service to Node N and possibly an *In* or a *Fault* message sent in the opposite direction.

AMDP used for this pattern are the same as in the Out-In pattern.

B.2.8 Additional MEPs

Since WSDL 2.0 allows new MEPs to be defined, each new MEP is expected to define how the AMDP feature is used in conjunction with that MEP. It is also possible to define a composite MEP that is composed of predefined MEPs. Such a composite MEP, when used in conjunction with AMDP, must not violate the AMDP rules defined for the predefined MEPs. For an example of such a composite MEP, see **5 CallBack Pattern** [p.31] that define Callback pattern.

B.2.9 Summary

The table below summarizes the properties used in the context of WSDL 2.0 MEPs and the restrictions on the value of those properties.

AMDP for WSDL 2.0 MEPs

		MO	MD	RD	FD	MID	MR	ON
In-Only		R	R (WSRef)	------	------	------	------	R
Robust In-Only	**Input**	R+	R (WSRef)	------	O (URI)	R	------	R
	Fault	R (WSRef)	R	------	------	------	R	R
In-Out	**Input**	R+	R (WSRef)	O (URI)	O (URI)	R	------	R
	Output	R (WSRef)	R	------	------	------	R	R
	Fault	R (WSRef)	R	------	------	------	R	R
Out-Only		R (WSRef)	R	------	------	------	------	R
Robust Out-Only	**Output**	R (WSRef)	R	------	O (URI)	R	------	R
	Fault	R (WSRef)	R	------	------	------	R	R
Out-In	**Output**	R (WSRef)	R	O (URI)	O (URI)	R	------	R
	Input	R	R (WSRef)	------	------	------	R	R
	Fault	R	R (WSRef)	------	------	------	R	R
Out-Optional-In	**Output**	R (WSRef)	R	O (URI)	O (URI)	------	------	R
	Input	R	R (WSRef)	------	------	------	R	R
	Fault	R	R (WSRef)	------	------	------	R	R

AMDP Acronyms:

MO = MessageOriginator; MD = MessageDestination; RD = ReplyDestination; FD = FaultDestination; MID = MessageID; MR = MessageReference; ON = OperationName

Value Designators:

R = Required; O = Optional; -- = Ignored; (...) = Type of the value; WSRef = WSDL 2.0 service element or QName; + = must be a URI if any of the optional non-ignored properties is absent

Red color = required value; Blue color = optional value;

C Message Delivery in a Mobile Context

Mobile handsets have become powerful enough in terms of processing power, memory, and network bandwidth to host Web browsers and viably supporting Web services applications. However, the adoption of Web service processing in mobile handsets is constrained by the cellular operator network.

Mobile phones do not typically have an IP address due to the limited number of IP addresses available in IPv4. Cellular operators use NATs and firewalls to allow mobile nodes to act as Internet clients only. To address a mobile handset using HTTP from the Internet is not a current deployment option. Hence, the current cellular data network is oriented to only enable simple, synchronous web services, initiated by the terminal. However, there is a suite of mobile applications which do not fit this simple programming model.

This specification defines programming patterns such as asynchronous request-response and callbacks that enable mobile terminals to use Web services to participate in more robust and complex mobile applications. In addition, mobile terminals will be capable of host Web services themselves. Services that encapsulate GPS location, calendar, or contact information are examples of such capabilities. Consequently, personal services such as these are dependent upon strict security and privacy constraints; their availability is highly constrained to limited authorized and authenticated clients.

A variety of mechanisms can be used to support these advanced mobile use cases. Two approaches are presented below:

- A foremost mechanism is to gateway through an intermediary capable of switching from HTTP to an appropriate mobile protocol such as SMS, MMS, or SIP. These mobile protocols alone, or in conjunction with a federated identity framework, have the capability to address the mobile terminal and transfer a SOAP message. For example, the address of the mobile terminal may be designated by using the `wsmd:ReplyDestination` property to indicate the uri of the gateway along with the data required for the intermediary to address the sending terminal, such as mobile phone number in the cases of SMS or MMS, or a URI in case of SIP.

- Another mechanism is to use a tunneling intermediary. Tunneling intermediaries are tightly bound to the mobile terminal - the mobile terminal initiates the connection and then keeps the HTTP session alive or reestablishing the connection when it breaks. The tunnel is URI addressable (i.e. john.doe.relay.example.org) and hence can be used as the value of the URI attribute that designates the destination, such as the value of the `wsmd:ReplyDestination` property. As the connection is always HTTP, there is no additional addressing data needed.

A signalling composite pattern is a very useful mechanism, particularly for message delivery in the mobile context. The signalling pattern can be described as follows.

1. A non-IP addressable device makes an initial request to a service. The device is not expecting a response right away.

2. The service, at a long or indeterminate time later, creates the results of the request. This may be large.

3. The service signals to the mobile device that the results of the request that are indicated in step 1 are available using SMS (a simple, limited, mobile protocol). The SMS signal includes data needed for correlation.

4. The mobile device receives the signal and correlates it with the initial request. The device then requests the response generated by the server.

This pattern is another example of a composite MEP which can be implemented using WS-MessageDelivery.

AMDPs may be used to realize this pattern. For example, the initial SOAP request in the first step would need to include an identification for the message sent that needs to be correlated later and a destination for the reply message. AMDP properties `wsmd:MessageID` and `wsmd:replyDestination` may be used to indicate the message identification and the URI of an SMS gateway respectively. The third step would utilize the identification that is communicated in the first step for correlation. At the fourth step, the mobile device may send an HTTP Get to receive the SOAP response to obtain the results that are generated at the second step. The URI for the GET is made explicit in the signal data. Alternatively, a SOAP Request/Response can be used to retrieve the Callback results by including the correlation in the request message, using the AMDP `wsmd:MessageReference`.