Web Services Make Connection (WS-MakeConnection) Version 1.0

Committee Specification

11 April 2007

Specification URIs:

This Version:
http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-cs-01.pdf
http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-cs-01.html
http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-cs-01.doc

Previous Version:
http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-cd-05.pdf
http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-cd-05.html
http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-cd-05.doc

Latest Version:
http://docs.oasis-open.org/ws-rx/wsmc/v1.0/wsmc.pdf
http://docs.oasis-open.org/ws-rx/wsmc/v1.0/wsmc.html
http://docs.oasis-open.org/ws-rx/wsmc/v1.0/wsmc.doc

Technical Committee:
OASIS Web Services Reliable Exchange (WS-RX) TC

Chairs:
Paul Fremantle <paul@wso2.com>
Sanjay Patil <sanjay.patil@sap.com>

Editors:
Doug Davis, IBM <doug@us.ibm.com>
Anish Karmarkar, Oracle <Anish.Karmarkar@oracle.com>
Gilbert Pilz, BEA <gpilz@bea.com>
Steve Winkler, SAP <steve.winkler@sap.com>
Ümit Yalçınalp, SAP <umit.yalcinalp@sap.com>

Declared XML Namespaces:
http://docs.oasis-open.org/ws-rx/wsmc/200702

Abstract:
This specification (WS-MakeConnection) describes a protocol that allows messages to be transferred between nodes implementing this protocol by using a transport-specific back-channel. The protocol is described in this specification in a transport-independent manner allowing it to be implemented using different network technologies. To support interoperable Web services, a SOAP binding is defined within this specification.

The protocol defined in this specification depends upon other Web services specifications for the identification of service endpoint addresses and policies. How these are identified and retrieved are detailed within those specifications and are out of scope for this document.

By using the XML [XML], SOAP [SOAP 1.1], [SOAP 1.2] and WSDL [WSDL 1.1] extensibility model, SOAP-based and WSDL-based specifications are designed to be composed with each other to define a rich Web services environment. As such, WS-MakeConnection by itself does not define all the
features required for a complete messaging solution. WS-MakeConnection is a building block that is used in conjunction with other specifications and application-specific protocols to accommodate a wide variety of requirements and scenarios related to the operation of distributed Web services.

**Status:**
This document was last revised or approved by the WS-RX Technical Committee on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee's web page at http://www.oasis-open.org/committees/ws-rx/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/ws-rx/ipr.php).

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/ws-rx/.
Notices

Copyright © OASIS® 1993–2007. All Rights Reserved. OASIS trademark, IPR and other policies apply.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS", WS-MakeConnection, WSMC, WSRM, WS-RX are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.
1 Introduction

The primary goal of this specification is to create a mechanism for the transfer of messages between two endpoints when the sending endpoint is unable to initiate a new connection to the receiving endpoint. It defines a mechanism to uniquely identify non-addressable endpoints, and a mechanism by which messages destined for those endpoints can be delivered. It also defines a SOAP binding that is required for interoperability. Additional bindings can be defined.

This mechanism is extensible allowing additional functionality, such as security, to be tightly integrated. This specification integrates with and complements the WS-ReliableMessaging[WS-RM], WS-Security [WS-Security], WS-Policy [WS-Policy], and other Web services specifications. Combined, these allow for a broad range of reliable, secure messaging options.

1.1 Terminology

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 [KEYWORDS].

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

- The syntax appears as an XML instance, but values in italics indicate data types instead of values.
- Characters are appended to elements and attributes to indicate cardinality:
  - "?" (0 or 1)
  - "*" (0 or more)
  - "+" (1 or more)
- The character "|" is used to indicate a choice between alternatives.
- The characters "[" and "]" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.
- An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content specified in this document. Additional children elements and/or attributes MAY be added at the indicated extension points but they MUST NOT contradict the semantics of the parent and/or owner, respectively. If an extension is not recognized it SHOULD be ignored.
- XML namespace prefixes (see section 1.4) are used to indicate the namespace of the element being defined.

Elements and Attributes defined by this specification are referred to in the text of this document using XPath 1.0 [XPATH 1.0] expressions. Extensibility points are referred to using an extended version of this syntax:

- An element extensibility point is referred to using {any} in place of the element name. This indicates that any element name can be used, from any namespace other than the wsmc: namespace.
- An attribute extensibility point is referred to using @{any} in place of the attribute name. This indicates that any attribute name can be used, from any namespace other than the wsmc: namespace.
1.2 Normative

http://www.ietf.org/rfc/rfc2119.txt

http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

http://ietf.org/rfc/rfc3986

http://www.ietf.org/rfc/rfc4122.txt

http://www.w3.org/TR/2001/NOTE-wsdl-20010315

http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/
http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/


http://www.w3.org/TR/REC-xml/

http://www.w3.org/TR/1999/REC-xml-names-19990114/

http://www.w3.org/TR/xmlschema-1/

http://www.w3.org/TR/xmlschema-2/

http://www.w3.org/TR/xpath

1.3 Non-Normative

http://www.openhealth.org/RDDL/20040118/rddl-20040118.html

wsmc-1.0-spec-cs-01
Copyright © OASIS® 1993–2007. All Rights Reserved. OASIS trademark, IPR and other policies apply. 11 April 2007 Page 6 of 27
1.4 Namespace

The XML namespace [XML-ns] URI that MUST be used by implementations of this specification is:

```
http://docs.oasis-open.org/ws-rx/wsmc/200702
```

Dereferencing the above URI will produce the Resource Directory Description Language [RDDL 2.0] document that describes this namespace.

Table 1 lists the XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant. The assertions defined within this specification have been designed to work independently of a specific version of WS-Policy and WS-Policy Attachment. Within this specification the use of the namespace prefix "wsp" refers generically to the WS-Policy namespace, not a specific version.

Table 1

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>(Either SOAP 1.1 or 1.2)</td>
</tr>
<tr>
<td>S11</td>
<td><a href="http://schemas.xmlsoap.org/soap/envelope/">http://schemas.xmlsoap.org/soap/envelope/</a></td>
</tr>
</tbody>
</table>
The normative schema for WS-MakeConnection can be found linked from the namespace document that is located at the namespace URI specified above.

All sections explicitly noted as examples are informational and are not to be considered normative.

### 1.5 Conformance

An implementation is not conformant with this specification if it fails to satisfy one or more of the MUST or REQUIRED level requirements defined herein. A SOAP Node MUST NOT use the XML namespace identifier for this specification (listed in section 1.4) within SOAP Envelopes unless it is conformant with this specification.

Normative text within this specification takes precedence over normative outlines, which in turn take precedence over the XML Schema [XML Schema Part 1, Part 2] descriptions.
2  MakeConnection Model

The WS-Addressing [WS-Addressing] specification defines the anonymous URI to identify non-addressable endpoints and to indicate a protocol-specific back-channel is to be used for any messages destined for that endpoint. For example, when used in the WS-Addressing ReplyTo EPR, the use of this anonymous URI is meant to indicate that any response message is to be transmitted on the transport-specific back-channel. In the HTTP case this would mean that any response message is sent back on the HTTP response flow.

In cases where the connection is still available the WS-Addressing URI is sufficient. However, in cases where the original connection is no longer available, additional mechanisms are needed. Take the situation where the original connection that carried a request message is broken and therefore is no longer available to carry a response back to the original sender. Traditionally, non-anonymous (addressable) EPRs would be used in these cases to allow for the sender of the response message to initiate new connections as needed. However, if the sender of the request message is unable (or unwilling) to accept new connections then the only option available is for it to establish a new connection for the purposes of allowing the response message to be sent. This specification defines a mechanism by which a new connection can be established.

The MakeConnection model consists of two key aspects:

- An optional anonymous-like URI template is defined that has similar semantics to WS-Addressing's anonymous, but also allows for each non-addressable endpoint to be uniquely identified
- A new message is defined that establishes a connection that can then be used to transmit messages to these non-addressable endpoints

Figure 1 below illustrates the overall flow involved in the use of MakeConnection:

![Make Connection Model Diagram]

The MakeConnection message is used to establish a new connection between the two endpoints. Within the message is identifying information that is used to uniquely identify a message that is eligible for transmission.
2.1 Glossary

The following definitions are used throughout this specification:

**Back-channel:** When the underlying transport provides a mechanism to return a transport-protocol specific response, capable of carrying a SOAP message, without initiating a new connection, this specification refers to this mechanism as a back-channel.

**Endpoint:** As defined in the WS-Addressing specification; a Web service Endpoint is a (referenceable) entity, processor, or resource to which Web service messages can be addressed. Endpoint references (EPRs) convey the information needed to address a Web service Endpoint.

**MC Initiator** The endpoint that transmits the MakeConnection message — the destination endpoint for the messages being sent on the transport-specific back-channel.

**MC Receiver:** The endpoint that receives the MakeConnection message — the source endpoint for the messages being sent on the transport-specific back-channel.

**Receive:** The act of reading a message from a network connection.

**Transmit:** The act of writing a message to a network connection.

2.2 Protocol Preconditions

The correct operation of the protocol requires that a number of preconditions MUST be established prior to the processing of the initial sequenced message:

- The MC Receiver MUST be capable of accepting new incoming connections.
- The MC Initiator MUST be capable of creating new outgoing connections to the MC Receiver, and those connections MUST have a back-channel.
- If a secure exchange of messages is REQUIRED, then the MC Initiator and MC Receiver MUST have a security context.

2.3 Example Message Exchange

Figure 2 illustrates a message exchange in which the response message is delivered using MakeConnection.
323 Figure 2: Example WS-MakeConnection Message Exchange

1. The protocol preconditions are established. These include policy exchange, endpoint resolution, and establishing trust.

2. The client (MC Initiator) sends a GetQuote request message to the service (MC Receiver). The WS-Addressing wsa:ReplyTo EPR uses the MakeConnection Anonymous URI Template – indicating that if the GetQuoteResponse message is not sent back on this connection's back-channel, then the client will use MakeConnection to retrieve it.

3. The service receives the request message and decides to close the connection by sending back an empty response (in the HTTP case an HTTP 202 Accept is sent).

4. The client sends a MakeConnection message to the service. Within the MakeConnection element is the wsmc:Address element containing the same MakeConnection Anonymous URI used in step 2.

5. The service has not completed executing the GetQuote operation and decides to close the connection by sending back an empty response (in the HTTP case an HTTP 202 Accept) indicating that no messages destined for this MC Initiator are available at this time.

6. The client sends a second MakeConnection message to the service. Within the MakeConnection element is the wsmc:Address element containing the same MakeConnection Anonymous URI used in step 2.

7. The service uses this new connection to transmit the GetQuoteResponse message.

The service can assume that because the MakeConnection Anonymous URI Template was used in the wsa:ReplyTo EPR the client will act as an MC Initiator for the purposes of retrieving messages destined to that EPR (i.e. responses to the GetQuote). This allows the service the option of immediately releasing resources used by the original connection – knowing that the client will, at some later point in time, establish a new connection on which the GetQuoteResponse can be transmitted. Likewise, when the first MakeConnection is received by the service, it again has the option of leaving the connection open until the GetQuoteResponse is ready to be transmitted, or it can close the connection immediately knowing that the MC Initiator will retransmit the MakeConnection message at some later point in time. Since the nature and dynamic characteristics of the underlying transport and potential intermediaries are unknown in the general case, the timing of re-transmissions cannot be specified. Additionally, over-aggressive re-transmissions have been demonstrated to cause transport or intermediary flooding which are counterproductive. Consequently, implementers are encouraged to utilize adaptive mechanisms that dynamically adjust re-
transmission time and the back-off intervals that are appropriate to the nature of the transports and intermediaries envisioned. For the case of TCP/IP transports, a mechanism similar to that described as RTTM in RFC 1323 [RTTM] SHOULD be considered.

Now that the basic model has been outlined, the details of this protocol are now provided in section 3.
3 MakeConnection

The following sub-sections define the various MakeConnection features, and prescribe their usage by a conformant implementations.

3.1 MakeConnection Anonymous URI

When an Endpoint is not directly addressable (e.g. behind a firewall or not able to allow incoming connections), an anonymous URI in the EPR address property can indicate such an Endpoint. The WS-Addressing anonymous URI is one such anonymous URI. This specification defines a URI template (the WS-MC anonymous URI) which may be used to uniquely identify anonymous Endpoints.

http://docs.oasis-open.org/ws-rx/wsmc/200702/anonymous?id={unique-String}

The appearance of an instance of this URI template in the wsa:Address value of an EPR indicates a protocol-specific back-channel will be established through a mechanism such as MakeConnection, defined below. When using this URI template, “{unique-String}” MUST be replaced by a globally unique string (e.g a UUID value as defined by RFC4122 [UUID]). This specification does not require the use of one particular string generation scheme. This string uniquely distinguishes the Endpoint. A sending Endpoint SHOULD Transmit messages at Endpoints identified with the URI template using a protocol-specific back-channel, including but not limited to those established with a MakeConnection message. Note, this URI template is semantically similar to the WS-Addressing anonymous URI if a protocol-specific back-channel is available.

3.2 MakeConnection Message

The MakeConnection element is sent in the body of a one-way message that establishes a contextualized back-channel for the transmission of messages according to matching criteria (defined below). In the non-faulting case, if no matching message is available then no SOAP envelope will be returned on the back-channel. A common usage will be a client sending MakeConnection to a server for the purpose of receiving asynchronous response messages.

The following exemplar defines the MakeConnection syntax:

```xml
<wsmc:MakeConnection ...>
    <wsmc:Address ...> xs:anyURI </wsmc:Address> ?
    <wsmc:Identifier ...> xs:anyURI </wsmc:Identifier> ?
    ...
</wsmc:MakeConnection>
```

The following describes the content model of the MakeConnection element.

/wsmc:MakeConnection

This element allows the sender to create a transport-specific back-channel that can be used to return a message that matches the selection criteria. Endpoints MUST NOT send this element as a header block. At least one selection criteria sub-element MUST be specified – if not a MissingSelection fault MUST be generated.

/wsmc:MakeConnection/wsmc:Address

This element specifies the URI (wsa:Address) of the initiating Endpoint. Endpoints MUST NOT return messages on the transport-specific back-channel unless they have been addressed to this URI. This Address property and a message’s WS-Addressing destination property are considered identical when they are exactly the same character-for-character. Note that URLs which are not
identical in this sense may in fact be functionally equivalent. Examples include URI references which differ only in case, or which are in external entities which have different effective base URLs.

/wsmc:MakeConnection/wsmc:Address/@{any}
This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the element.

/wsmc:MakeConnection/wsrn:Identifier
This element specifies the WS-RM Sequence Identifier that establishes the context for the transport-specific back-channel. The Sequence Identifier should be compared with the Sequence Identifiers associated with the messages held by the sending Endpoint, and if there is a matching message it will be returned.

/wsmc:MakeConnection/wsrn:Identifier/@{any}
This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the element.

/wsmc:MakeConnection/{any}
This is an extensibility mechanism to allow different (extensible) types of information, based on a schema, to be passed. This allows fine-tuning of the messages to be returned, additional selection criteria included here are logically ANDed with the Address and/or wsrn:Identifier. If an extension is not supported by the Endpoint then it should generate an UnsupportedSelection fault.

/wsmc:MakeConnection/@{any}
This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the element.

If more than one selection criteria element is present, then the MC Receiver processing the MakeConnection message MUST insure that any SOAP Envelope flowing on the back-channel satisfies all of those selection criteria.

The management of messages that are awaiting the establishment of a back-channel to their receiving Endpoint is an implementation detail that is outside the scope of this specification. Note, however, that these messages form a class of asynchronous messages that is not dissimilar from "ordinary" asynchronous messages that are waiting for the establishment of a connection to their destination Endpoints.

This specification places no constraint on the types of messages that can be returned on the transport-specific back-channel. As in an asynchronous environment, it is up to the recipient of the MakeConnection message to decide which messages are appropriate for transmission to any particular Endpoint. However, the Endpoint processing the MakeConnection message MUST insure that the messages match the selection criteria as specified by the child elements of the MakeConnection element.

Since the message exchange pattern use by MakeConnection is untraditional, the following points need to be reiterated for clarification:

- The MakeConnection message is logically part of a one-way operation; there is no reply message to the MakeConnection itself, and any response flowing on the transport back-channel is a pending message.
- Since there is no reply message to MakeConnection, the WS-Addressing specific rules in section 3.4 "Formulating a Reply Message" are not used. Therefore, the value of any wsa:ReplyTo element in the MakeConnection message has no effective impact since the WS-
Addressing [reply endpoint] property that is set by the presence of wsa:ReplyTo is not used.

- In the absence of any pending message, there will be no message transmitted on the transport back-channel. E.g. in the HTTP case just an HTTP 202 Accepted will be returned without any SOAP envelope in the HTTP response message.

- When there is a message pending, it is sent on the transport back-channel, using the connection that has been initiated by the MakeConnection request.

### 3.3 MessagePending

When MakeConnection is used, and a message is returned on the transport-specific back-channel, the MessagePending header SHOULD be included on the returned message as an indicator whether there are additional messages waiting to be retrieved using the same selection criteria that was specified in the MakeConnection element.

The following exemplar defines the MessagePending syntax:

```xml
<wsmc:MessagePending pending="xs:boolean" ...>
  ...
</wsmc:MessagePending>
```

The following describes the content model of the MessagePending header block.

//wsmc:MessagePending

This element indicates whether additional messages are waiting to be retrieved.

//wsmc:MessagePending/@pending

This attribute, when set to "true", indicates that there is at least one message waiting to be retrieved. When this attribute is set to "false" it indicates there are currently no messages waiting to be retrieved.

//wsmc:MessagePending/{any}

This is an extensibility mechanism to allow different (extensible) types of information, based on a schema, to be passed.

//wsmc:MessagePending/@{any}

This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the element.

The absence of the MessagePending header has no implication as to whether there are additional messages waiting to be retrieved.

### 3.4 MakeConnection Policy Assertion

The MakeConnection policy assertion indicates that the MakeConnection protocol (operation and the use of the MakeConnection URI template in EndpointReferences) is required for messages sent from this endpoint. This assertion has Endpoint Policy Subject [WS-PolicyAttachment].

The normative outline for the MakeConnection assertion is:

```xml
<wsmc:MCSupported ...> ... </wsmc:MCSupported>
```

The following describes the content model of the MCSupported element.
A policy assertion that specifies that the MakeConnection protocol is required for messages sent from this endpoint.

/wsmc:MCSupported/{any}
This is an extensibility mechanism to allow different (extensible) types of information, based on a schema, to be passed.

/wsmc:MCSupported/@{any}
This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the element.
4 Faults

Entities that generate WS-MakeConnection faults MUST include as the [action] property the default fault action IRI defined below. The value from the W3C Recommendation is below for informational purposes:

```
http://docs.oasis-open.org/ws-rx/wsmc/200702/fault
```

The faults defined in this section are generated if the condition stated in the preamble is met. Fault handling rules are defined in section 6 of WS-Addressing SOAP Binding.

The definitions of faults use the following properties:

- [Code] The fault code.
- [Subcode] The fault subcode.
- [Detail] The detail element(s). If absent, no detail element is defined for the fault. If more than one detail element is defined for a fault, implementations MUST include the elements in the order that they are specified.

Entities that generate WS-MakeConnection faults MUST set the [Code] property to either "Sender" or "Receiver". These properties are serialized into text XML as follows:

<table>
<thead>
<tr>
<th>SOAP Version</th>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP 1.1</td>
<td>S11:Client</td>
<td>S11:Server</td>
</tr>
<tr>
<td>SOAP 1.2</td>
<td>S:Sender</td>
<td>S:Receiver</td>
</tr>
</tbody>
</table>

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

```
<S:Envelope>
  <S:Header>
    <wsa:Action>
      http://docs.oasis-open.org/ws-rx/wsmc/200702/fault
    </wsa:Action>
  </S:Header>
  <S:Body>
    <S:Fault>
      <S:Code>
        <S:Value>[Code]</S:Value>
        <S:Subcode>
          <S:Value>[Subcode]</S:Value>
        </S:Subcode>
      </S:Code>
      <S:Reason>
        <S:Text xml:lang="en">[Reason]</S:Text>
      </S:Reason>
      <S:Detail>
        <Detail>
        ...  
      </S:Detail>
    </S:Fault>
    <S:Body>
    </S:Body>
  </S:Body>
</S:Envelope>
```

The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a result of processing a MakeConnection message:
4.1 Unsupported Selection

The QName of the unsupported element(s) are included in the detail.

Properties:

[Code] Receiver

[Subcode] wsmc:UnsupportedSelection

[Reason] The extension element used in the message selection is not supported by the MakeConnection receiver

[Detail]

<wsmc:UnsupportedSelection> xs:QName </wsmc:UnsupportedSelection>+

<table>
<thead>
<tr>
<th>Generated by</th>
<th>Condition</th>
<th>Action Upon Generation</th>
<th>Action Upon Receipt</th>
</tr>
</thead>
<tbody>
<tr>
<td>MakeConnection receiver</td>
<td>In response to a MakeConnection message containing a selection criteria in the extensibility section of the message that is not supported</td>
<td>Unspecified.</td>
<td>Unspecified.</td>
</tr>
</tbody>
</table>

4.2 Missing Selection

The MakeConnection element did not contain any selection criteria.

Properties:

[Code] Receiver

[Subcode] wsmc:MissingSelection

[Reason] The MakeConnection element did not contain any selection criteria.

[Detail]
<table>
<thead>
<tr>
<th>Generated by</th>
<th>Condition</th>
<th>Action Upon Generation</th>
<th>Action Upon Receipt</th>
</tr>
</thead>
<tbody>
<tr>
<td>MakeConnection receiver</td>
<td>In response to a MakeConnection message that does not contain any selection criteria</td>
<td>Unspecified.</td>
<td>Unspecified.</td>
</tr>
</tbody>
</table>
5 Security Considerations

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

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

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

Recipients of messages are RECOMMENDED to validate the signature to authenticate and verify the integrity of the data. Specifically, recipients SHOULD verify that the sender has the right to "speak" for the message.

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

- Message alteration - Alteration is prevented by including signatures of the message information using WS-Security.
- Message disclosure - Confidentiality is preserved by encrypting sensitive data using WS-Security.
- Key integrity - Key integrity is maintained by using the strongest algorithms possible (by comparing secured policies - see WS-Policy and WS-SecurityPolicy [SecurityPolicy]).
- Authentication - Authentication is established using the mechanisms described in WS-Security and WS-Trust. Each message is authenticated using the mechanisms described in WS-Security.
- Accountability - Accountability is a function of the type of and strength of the key and algorithms being used. In many cases, a strong symmetric key provides sufficient accountability. However, in some environments, strong PKI signatures are required.
- Availability - All reliable messaging services are subject to a variety of availability attacks. Replay detection is a common attack and it is RECOMMENDED that this be addressed by the mechanisms described in WS-Security. Other attacks, such as network-level denial of service attacks are harder to avoid and are outside the scope of this specification. That said, care should be taken to ensure that minimal state is saved prior to any authenticating sequences.
- Replay - Messages may be replayed for a variety of reasons. To detect and eliminate this attack, mechanisms should be used to identify replayed messages such as the timestamp/nonce outlined in WS-Security. Alternatively, and optionally, other technologies, such as sequencing, can also be used to prevent replay of application messages.

Service endpoints SHOULD scope its searching of messages to those that were processed under the same security context as the requesting MakeConnection message.
The normative schema that is defined for WS-MakeConnection using [XML-Schema Part1] and [XML-Schema Part2] is located at:

http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-schema-200702.xsd

The following copy is provided for reference.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (C) OASIS(R) 1993-2007. All Rights Reserved. OASIS trademark, IPR and other policies apply. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
  targetNamespace="http://docs.oasis-open.org/ws-rx/wsmc/200702"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:import namespace="http://www.w3.org/2005/08/addressing"
    schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
  <!-- Protocol Elements -->
  <xs:complexType name="MessagePendingType">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="pending" type="xs:boolean"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:element name="MessagePending" type="wsmc:MessagePendingType"/>
  <xs:element name="Address">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:anyURI">
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="MakeConnectionType">
    <xs:sequence>
      <xs:element ref="wsmc:Address" minOccurs="0" maxOccurs="1"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="MakeConnection" type="wsmc:MakeConnectionType"/>
  <xs:element name="UnsupportedSelection">
    <xs:simpleType>
      <xs:restriction base="xs:QName"/>
    </xs:simpleType>
  </xs:element>
</xs:schema>
```
Appendix B. WSDL

This WSDL describes the WS-MC protocol from the point of view of the endpoint that receives the MakeConnection message.

Also note that this WSDL is intended to describe the internal structure of the WS-MC protocol, and will not generally appear in a description of a WS-MC-capable Web service. See section 3.4 Policy for a higher-level mechanism to indicate that WS-MC is supported.

The normative WSDL 1.1 definition for WS-MakeConnection is located at:

http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-wsdl-200702.wsdl

The following non-normative copy is provided for reference.

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
OASIS trademark, IPR and other policies apply. -->
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:wsam="http://www.w3.org/2007/02/addressing/metadata"
xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
xmlns:tns="http://docs.oasis-open.org/ws-rx/wsmc/200702/wsdl"
targetNamespace="http://docs.oasis-open.org/ws-rx/wsmc/200702/wsdl">
<wsdl:types>
  <xs:schema>
  </xs:schema>
</wsdl:types>
<wsdl:message name="MakeConnection">
  <wsdl:part name="makeConnection" element="wsmc:MakeConnection"/>
</wsdl:message>
<wsdl:portType name="MCAbstractPortType">
  <wsdl:operation name="MakeConnection">
    <wsdl:input message="tns:MakeConnection" wsam:Action="http://docs.oasis-open.org/ws-rx/wsmc/200702/MakeConnection"/>
    <!-- As described in the WS-MakeConnection specification, the MakeConnection operation establishes a connection. If a matching message is available then the back-channel of the connection will be used to carry the message. In SOAP terms the returned message is not a response, so there is no WSDL output message. -->
  </wsdl:operation>
</wsdl:portType>
</wsdl:definitions>
```
Appendix C. Message Examples

Appendix C.1 Example use of MakeConnection

To illustrate how a MakeConnection message exchange can be used to deliver messages to an Endpoint that is not addressable, consider the case of a pub/sub scenario in which the Endpoint to which notifications are to be delivered (the "event consumer") is not addressable by the notification sending Endpoint (the "event producer"). In this scenario the event consumer must initiate the connections in order for the notifications to be delivered. One possible set of message exchanges (using HTTP) that demonstrate how this can be achieved using MakeConnection is shown below.

Step 1 – During a "subscribe" operation, the event consumer's EPR specifies the MC anonymous URI and the WS-RM Policy Assertion [WS-RM Policy] to indicate whether or not RM is required:

```xml
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
  xmlns:wsrmp="http://docs.oasis-open.org/ws-rx/wsrmp/200702"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:To> http://example.org/subscriptionService </wsa:To>
    <wsa:MessageID> http://client456.org/id-a6d8-a7c2eb546813 </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:To> http://client456.org/response </wsa:To>
    </wsa:ReplyTo>
  </S:Header>
  <S:Body>
    <sub:Subscribe xmlns:sub="http://example.org/subscriptionService">
      <!-- subscription service specific data -->
      <targetEPR>
        <wsa:Address>http://docs.oasis-open.org/ws-rx/wsrmp/200702/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:Address>
        <wsa:Metadata>
          <wsp:Policy wsu:Id="MyPolicy">
            <wsrmp:RMAssertion/>
          </wsp:Policy>
        </wsa:Metadata>
      </targetEPR>
    </sub:Subscribe>
  </S:Body>
</S:Envelope>
```

In this example the subscribe and targetEPR elements are simply examples of what a subscription request message might contain. Note: the wsa:Address element contains the MC anonymous URI indicating that the notification producer needs to queue the messages until they are requested using the MakeConnection message exchange. The EPR also contains the WS-RM Policy Assertion indicating the RM must be used when notifications related to this subscription are sent.

Step 2 – Once the subscription is established, the event consumer checks for a pending message:

```xml
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
  </S:Header>
  <wsa:To> http://example.org/subscriptionService </wsa:To>
</S:Envelope>
```
Step 3 – If there are messages waiting to be delivered then a message will be returned back to the event consumer. However, because WS-RM is being used to deliver the messages, the first message returned is a CreateSequence:

```xml
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
    xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200702"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:To>http://docs.oasis-open.org/ws-rx/wsrm/200702/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:To>
    <wsa:ReplyTo>http://example.org/subscriptionService</wsa:ReplyTo>
    <wsa:MessageID>http://example.org/id-123-456</wsa:MessageID>
    <wsmc:MessagePending pending="true"/>
  </S:Header>
  <S:Body>
    <wsrm:CreateSequence>
      <wsrm:AcksTo>
        <wsa:Address>http://example.org/subscriptionService</wsa:Address>
      </wsrm:AcksTo>
    </wsrm:CreateSequence>
  </S:Body>
</S:Envelope>
```

Notice from the perspective of how the RM Source on the event producer interacts with the RM Destination of those messages, nothing new is introduced by the use of the MakeConnection, the use of RM protocol is the same as the case where the event consumer is addressable. Note the message contains a wsmc:MessagePending header indicating that additional message are waiting to be delivered.

Step 4 – The event consumer will respond with a CreateSequenceResponse message per normal WS-Addressing rules:

```xml
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200702"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:To>http://example.org/subscriptionService</wsa:To>
    <wsa:RelatesTo>http://example.org/id-123-456</wsa:RelatesTo>
  </S:Header>
  <S:Body>
    <wsrm:CreateSequenceResponse>
      <wsrm:Identifier>http://example.org/rmid-456</wsrm:Identifier>
    </wsrm:CreateSequenceResponse>
  </S:Body>
</S:Envelope>
```
Note, this message is carried on an HTTP request directed to the wsa:ReplyTo EPR, and the HTTP response will be an HTTP 202.

Step 5 – The event consumer checks for another message pending:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:To>http://example.org/subscriptionService</wsa:To>
  </S:Header>
  <S:Body>
    <wsmc:MakeConnection>
      <wsmc:Address>http://docs.oasis-open.org/ws-rx/wsmc/200702/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsmc:Address>
    </wsmc:MakeConnection>
  </S:Body>
</S:Envelope>
```

Notice this is the same message as the one sent in step 2.

Step 6 – Since there is a message pending for this destination then it is returned on the HTTP response:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:Action>http://example.org/eventType1</wsa:Action>
    <wsa:To>http://docs.oasis-open.org/ws-rx/wsmc/200702/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:To>
    <wsrm:Sequence>
      <wsrm:Identifier>http://example.org/rmid-456</wsrm:Identifier>
    </wsrm:Sequence>
    <wsmc:MessagePending pending="true"/>
  </S:Header>
  <S:Body>
    <!-- event specific data -->
  </S:Body>
</S:Envelope>
```

As noted in step 3, the use of the RM protocol does not change when using MakeConnection. The format of the messages, the order of the messages sent and the timing of when to send it remains the same.

Step 7 – At some later interval, or immediately due to the MessagePending header's "pending" attribute being set to "true", the event consumer will poll again:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:To>http://example.org/subscriptionService</wsa:To>
  </S:Header>
</S:Envelope>
```
Notice this is the same message as the one sent in steps 2 and 5. As in steps 3 and 6, the response to the `MakeConnection` can be any message destined to the specified Endpoint. This allows the event producer to send not only application messages (events) but RM protocol messages (e.g. `CloseSequence`, `TerminateSequence` or even additional `CreateSequence` messages) as needed.

Step 8 – If at any point in time there are no messages pending, in response to a `MakeConnection` the event producer returns an HTTP 202 back to the event consumer. The process then repeats (back to step 7) until the subscription ends.
Appendix D. Acknowledgments

The following individuals have provided invaluable input into the initial contribution:

866  Keith Ballinger, Microsoft                      880  Efim Hudis, Microsoft
867  Stefan Batres, Microsoft                      881  David Ingham, Microsoft
868  Rebecca Bergersen, Iona                      882  Gopal Kakivaya, Microsoft
869  Allen Brown, Microsoft                       883  Johannes Klein, Microsoft
870  Kyle Brown, IBM                                884  Frank Leymann, IBM
871  Michael Conner, IBM                           885  Martin Nally, IBM
872  George Copeland, Microsoft                    886  Peter Niblett, IBM
873  Francisco Curbera, IBM                        887  Jeffrey Schlimmer, Microsoft
874  Paul Fremantle, IBM                            888  James Snell, IBM
875  Steve Graham, IBM                             889  Keith Stobie, Microsoft
876  Pat Helland, Microsoft                        890  Satish Thatte, Microsoft
877  Rick Hill, Microsoft                          891  Stephen Todd, IBM
878  Scott Hinkelman, IBM                          892  Sanjiva Weerawarana, IBM
879  Tim Holloway, IBM                             893  Roger Wolter, Microsoft

894  The following individuals were members of the committee during the development of this specification:

895  Abbie Barbir, Nortel                            925  Paul Knight, Nortel
896  Charlton Barreto, Adobe                        926  Dan Leshchiner, Tibco
897  Stefan Batres, Microsoft                       927  Mark Little, JBoss
898  Hamid Ben Malek, Fujitsu                      928  Lily Liu, webMethods
899  Andreas Bjarlestad, Ericsson                   929  Matt Lovett, IBM
900  Toufic Boubez, Layer 7                         930  Ashok Malhotra, Oracle
901  Doug Bunting, Sun                              931  Jonathan Marsh, Microsoft
902  Lloyd Burch, Novell                            932  Daniel Millwood, IBM
903  Steve Carter, Novell                           933  Jeff Mischkinsky, Oracle
904  Martin Chapman, Oracle                        934  Nilo Mitra, Ericsson
905  Dave Chappell, Sonic                           935  Peter Niblett, IBM
906  Paul Cotton, Microsoft                         936  Duane Nickull, Adobe
907  Glen Daniels, Sonic                            937  Eisaku Nishiyama, Hitachi
908  Doug Davis, IBM                                938  Dave Orchard, BEA
909  Blake Dournae, Intel                           939  Chouthri Palanisamy, NEC
910  Jacques Durand, Fujitsu                        940  Sanjay Patil, SAP
911  Colleen Evans, Microsoft                       941  Gilbert Pilz, BEA
912  Christopher Ferris, IBM                        942  Martin Raepple, SAP
913  Paul Fremantle, WSO2                           943  Eric Rajkovic, Oracle
914  Robert Freund, Hitachi                         944  Stefan Rossmannith, SAP
915  Peter Furniss, Erebor                          945  Tom Rutt, Fujitsu
916  Marc Goodner, Microsoft                        946  Rich Salz, IBM
917  Alastair Green, Choreology                    947  Shivajee Samdarshi, Tibco
918  Mike Grogan, Sun                               948  Vladimir Vidalov, SAP
919  Ondrej Hrebicek, Microsoft                     949  Claus von Riegen, SAP
920  Kazunori Iwasa, Fujitsu                       950  Pete Wenzel, Sun
921  Chamikara Jayalatha, WSO2                      951  Steve Winkler, SAP
922  Lei Jin, BEA                                   952  Ümit Yalçınalp, SAP
923  Ian Jones, BTplc                                953  Nobuyuki Yamamoto, Hitachi
924  Anish Karmarkar, Oracle                        954