



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

Web Services Make Connection (WS-MakeConnection) Version 1.0

OASIS Standard incorporating Approved Errata

07 January 2008

Specification URIs:

This Version:

- <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-os-01-e1.pdf>
- <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-os-01-e1.html>
- <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-os-01-e1.doc>

Previous Version:

- <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-os-01.pdf>
- <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-os-01.html>
- <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-os-01.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 <dug@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

44 features required for a complete messaging solution. WS-MakeConnection is a building block that is
45 used in conjunction with other specifications and application-specific protocols to accommodate a
46 wide variety of requirements and scenarios related to the operation of distributed Web services.

47 **Status:**

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

51 Technical Committee members should send comments on this specification to the Technical
52 Committee's email list. Others should send comments to the Technical Committee by using the "Send
53 A Comment" button on the Technical Committee's web page at [http://www.oasis-
54 open.org/committees/ws-rx/](http://www.oasis-open.org/committees/ws-rx/).

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

59 The non-normative errata page for this specification is located at [http://www.oasis-
60 open.org/committees/ws-rx/](http://www.oasis-open.org/committees/ws-rx/).

61 Notices

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

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

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

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

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

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

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

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

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

106 Table of Contents

107	1	Introduction	5
108	1.1	Terminology	5
109	1.2	Normative	6
110	1.3	Non-Normative	6
111	1.4	Namespace	7
112	1.5	Conformance	8
113	2	MakeConnection Model	9
114	2.1	Glossary	10
115	2.2	Protocol Preconditions	10
116	2.3	Example Message Exchange	10
117	3	MakeConnection	13
118	3.1	MakeConnection Anonymous URI	13
119	3.2	MakeConnection Message	13
120	3.3	MessagePending	15
121	3.4	MakeConnection Policy Assertion	15
122	4	Faults	17
123	4.1	Unsupported Selection	18
124	4.2	Missing Selection	18
125	5	Security Considerations	20
126		Appendix A. Schema	21
127		Appendix B. WSDL	22
128		Appendix C. Message Examples	23
129		Appendix C.1 Example use of MakeConnection	23
130		Appendix D. Acknowledgments	27
131			

132 1 Introduction

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

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

142 1.1 Terminology

143 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
144 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described
145 in RFC 2119 [[KEYWORDS](#)].

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

- 147 • The syntax appears as an XML instance, but values in italics indicate data types instead of values.
- 148 • Characters are appended to elements and attributes to indicate cardinality:
 - 149 ○ "?" (0 or 1)
 - 150 ○ "*" (0 or more)
 - 151 ○ "+" (1 or more)
- 152 • The character "|" is used to indicate a choice between alternatives.
- 153 • The characters "[" and "]" are used to indicate that contained items are to be treated as a group
154 with respect to cardinality or choice.
- 155 • An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content
156 specified in this document. Additional children elements and/or attributes MAY be added at the
157 indicated extension points but they MUST NOT contradict the semantics of the parent and/or
158 owner, respectively. If an extension is not recognized it SHOULD be ignored.
- 159 • XML namespace prefixes (see section 1.4) are used to indicate the namespace of the element
160 being defined.

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

- 164 • An element extensibility point is referred to using {any} in place of the element name. This
165 indicates that any element name can be used, from any namespace other than the `wsmc:`
166 namespace.
- 167 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This
168 indicates that any attribute name can be used, from any namespace other than the `wsmc:`
169 namespace.

170 1.2 Normative

- 171 **[KEYWORDS]** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC
172 2119, Harvard University, March 1997
173 <http://www.ietf.org/rfc/rfc2119.txt>
- 174 **[SOAP 1.1]** W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
175 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 176 **[SOAP 1.2]** W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework" June
177 2003.
178 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
- 179 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI):
180 Generic Syntax," RFC 3986, MIT/LCS, U.C. Irvine, Xerox Corporation, January
181 2005.
182 <http://ietf.org/rfc/rfc3986>
- 183 **[UUID]** P. Leach, M. Mealling, R. Salz, "A Universally Unique Identifier (UUID) URN
184 Namespace," RFC 4122, Microsoft, Refactored Networks - LLC, DataPower
185 Technology Inc, July 2005
186 <http://www.ietf.org/rfc/rfc4122.txt>
- 187 **[WSDL 1.1]** W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.
188 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 189 **[WS-Addressing]** W3C Recommendation, "Web Services Addressing 1.0 - Core", May 2006.
190 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
191 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding", May
192 2006.
193 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>
- 194 **[WS-RM]** OASIS WS-RX Technical OASIS Standard, "Web Services Reliable Messaging
195 (WS-ReliableMessaging)," June 2007.
196 <http://docs.oasis-open.org/ws-rx/wsr/v1.1/wsrp.pdf>
- 197 **[WS-RM Policy]** OASIS WS-RX Technical OASIS Standard, "Web Services Reliable Messaging
198 Policy Assertion(WS-RM Policy)" June 2007
199 <http://docs.oasis-open.org/ws-rx/wsrp/v1.1/wsrmp.pdf>
- 200 **[XML]** W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth
201 Edition)", September 2006.
202 <http://www.w3.org/TR/REC-xml/>
- 203 **[XML-ns]** W3C Recommendation, "Namespaces in XML," 14 January 1999.
204 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
- 205 **[XML-Schema Part1]** W3C Recommendation, "XML Schema Part 1: Structures," October 2004.
206 <http://www.w3.org/TR/xmlschema-1/>
- 207 **[XML-Schema Part2]** W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.
208 <http://www.w3.org/TR/xmlschema-2/>
- 209 **[XPath 1.0]** W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16 November
210 1999.
211 <http://www.w3.org/TR/xpath>

212 1.3 Non-Normative

- 213 **[RDDL 2.0]** Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language
214 (RDDL) 2.0," January 2004
215 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>

216 **[RTTM]** V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance",
 217 RFC 1323, May 1992.
 218 <http://www.rfc-editor.org/rfc/rfc1323.txt>

219 **[SecurityPolicy]** G. Della-Libra, et. al. "Web Services Security Policy Language (WS-
 220 SecurityPolicy)", July 2005
 221 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>

222 **[SecureConversation]** S. Anderson, et al, "Web Services Secure Conversation Language (WS-
 223 SecureConversation)," February 2005.
 224 <http://schemas.xmlsoap.org/ws/2004/04/sc/>

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

227 **[WS-Policy]** W3C Member Submission "Web Services Policy 1.2 - Framework", April 2006
 228 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>
 229
 230 W3C Recommendation, "Web Services Policy 1.5 - Framework," September
 231 2007.
 232 <http://www.w3.org/TR/2007/REC-ws-policy-20070904>

233 **[WS-PolicyAttachment]** W3C Member Submission "Web Services Policy 1.2 - Attachment", April
 234 2006
 235 <http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/>
 236
 237 W3C Recommendation, "Web Services Policy 1.5 - Attachment," September
 238 2007.
 239 <http://www.w3.org/TR/2007/REC-ws-policy-attach-2007004>

240 **[WS-Security]** Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS
 241 Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)",
 242 OASIS Standard 200401, March 2004.
 243 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
 244 [security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)

245
 246 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS
 247 Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", OASIS
 248 Standard 200602, February 2006.
 249 <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

250 1.4 Namespace

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

252 <http://docs.oasis-open.org/ws-rx/wsmc/200702>

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

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

260 Table 1

Prefix	Namespace
S	(Either SOAP 1.1 or 1.2)
S11	http://schemas.xmlsoap.org/soap/envelope/

S12	http://www.w3.org/2003/05/soap-envelope
wsmc	http://docs.oasis-open.org/ws-rx/wsmc/200702
wstrm	http://docs.oasis-open.org/ws-rx/wstrm/200702
wsa	http://www.w3.org/2005/08/addressing
wsam	http://www.w3.org/2007/05/addressing/metadata
xs	http://www.w3.org/2001/XMLSchema

261 The normative schema for WS-MakeConnection can be found linked from the namespace document that
 262 is located at the namespace URI specified above.

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

264 1.5 Conformance

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

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

271 2 MakeConnection Model

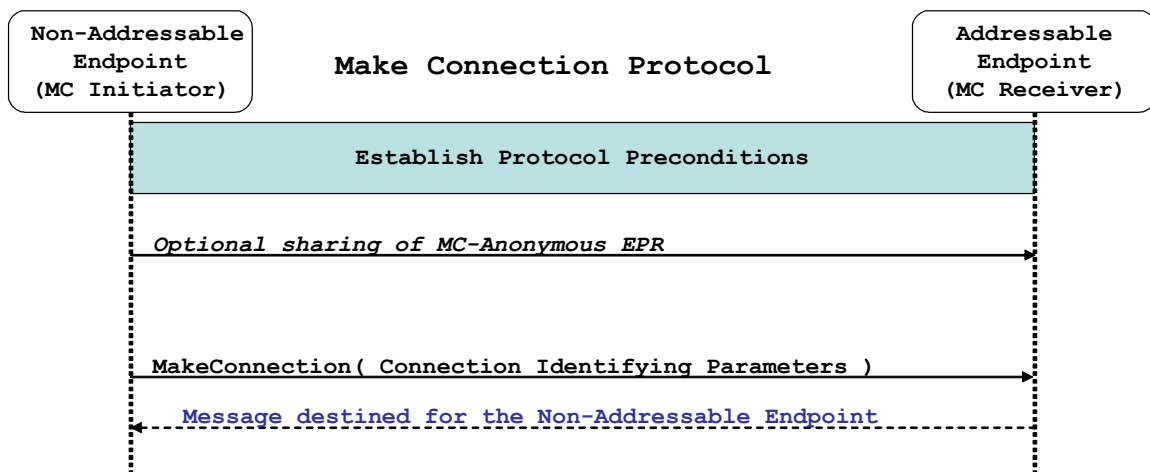
272 The WS-Addressing [WS-Addressing] specification defines the anonymous URI to identify non-
273 addressable endpoints and to indicate a protocol-specific back-channel is to be used for any messages
274 destined for that endpoint. For example, when used in the WS-Addressing ReplyTo EPR, the use of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

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

287 The MakeConnection model consists of two key aspects:

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

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



294 Figure 1 – Make Connection Model

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

298 **2.1 Glossary**

299 The following definitions are used throughout this specification:

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

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

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

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

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

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

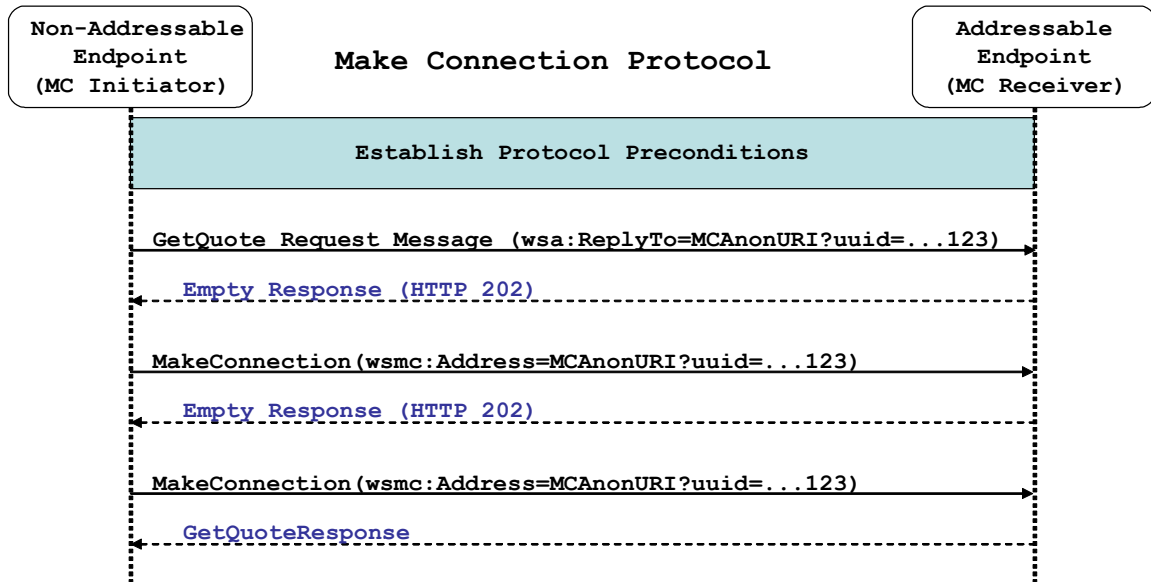
312 **2.2 Protocol Preconditions**

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

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

320 **2.3 Example Message Exchange**

321 Figure 2 illustrates a message exchange in which the response message is delivered using
322 `MakeConnection`.



323 Figure 2: Example WS-MakeConnection Message Exchange

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

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

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

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

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

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

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

342 The service can assume that because the `MakeConnection` Anonymous URI Template was used in the
343 `wsa:ReplyTo` EPR the client will act as an MC Initiator for the purposes of retrieving messages destined
344 to that EPR (i.e. responses to the `GetQuote`). This allows the service the option of immediately releasing
345 resources used by the original connection – knowing that the client will, at some later point in time,
346 establish a new connection on which the `GetQuoteResponse` can be transmitted. Likewise, when the first
347 `MakeConnection` is received by the service, it again has the option of leaving the connection open until
348 the `GetQuoteResponse` is ready to be transmitted, or it can close the connection immediately knowing
349 that the MC Initiator will retransmit the `MakeConnection` message at some later point in time. Since the
350 nature and dynamic characteristics of the underlying transport and potential intermediaries are unknown in
351 the general case, the timing of re-transmissions cannot be specified. Additionally, over-aggressive re-
352 transmissions have been demonstrated to cause transport or intermediary flooding which are
353 counterproductive. Consequently, implementers are encouraged to utilize adaptive mechanisms that

354 dynamically adjust re-transmission time and the back-off intervals that are appropriate to the nature of the
355 transports and intermediaries envisioned. For the case of TCP/IP transports, a mechanism similar to that
356 described as RTTM in RFC 1323 [[RTTM](#)] SHOULD be considered.

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

358 3 MakeConnection

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

361 3.1 MakeConnection Anonymous URI

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

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

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

376 3.2 MakeConnection Message

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

382 The following exemplar defines the `MakeConnection` syntax:

```
383 <wsmc:MakeConnection ...>  
384   <wsmc:Address ...> xs:anyURI </wsmc:Address> ?  
385   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier> ?  
386   ...  
387 </wsmc:MakeConnection>
```

388 The following describes the content model of the `MakeConnection` element.

389 `/wsmc:MakeConnection`

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

394 `/wsmc:MakeConnection/wsmc:Address`

395 This element specifies the URI (`wsa:Address`) of the initiating Endpoint. Endpoints MUST NOT
396 return messages on the transport-specific back-channel unless they have been addressed to this
397 URI. This Address property and a message’s WS-Addressing destination property are considered
398 identical when they are exactly the same character-for-character. Note that URIs which are not

399 identical in this sense may in fact be functionally equivalent. Examples include URI references
400 which differ only in case, or which are in external entities which have different effective base URIs.

401 `/wsmc:MakeConnection/wsmc:Address/@{any}`

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

404 `/wsmc:MakeConnection/wsrn:Identifier`

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

409 `/wsmc:MakeConnection/wsrn:Identifier/@{any}`

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

412 `/wsmc:MakeConnection/{any}`

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

418 `/wsmc:MakeConnection/@{any}`

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

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

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

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

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

- 437 • The `MakeConnection` message is logically part of a one-way operation; there is no reply
438 message to the `MakeConnection` itself, and any response flowing on the transport back-channel
439 is a pending message.
- 440 • Since there is no reply message to `MakeConnection`, the WS-Addressing specific rules in
441 section 3.4 "Formulating a Reply Message" are not used. Therefore, the value of any
442 `wsa:ReplyTo` element in the `MakeConnection` message has no effective impact since the WS-

- 443 Addressing [reply endpoint] property that is set by the presence of `wsa:ReplyTo` is not
444 used.
- 445 • In the absence of any pending message, there will be no message transmitted on the transport
446 back-channel. E.g. in the HTTP case just an `HTTP 202 Accepted` will be returned without any
447 SOAP envelope in the HTTP response message.
 - 448 • When there is a message pending, it is sent on the transport back-channel, using the connection
449 that has been initiated by the `MakeConnection` request.

450 3.3 MessagePending

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

455 The following exemplar defines the `MessagePending` syntax:

```
456 <wsmc:MessagePending pending="xs:boolean" ...>  
457   ...  
458 </wsmc:MessagePending>
```

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

460 `/wsmc:MessagePending`

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

462 `/wsmc:MessagePending/@pending`

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

466 `/wsmc:MessagePending/{any}`

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

469 `/wsmc:MessagePending/@{any}`

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

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

474 3.4 MakeConnection Policy Assertion

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

478 The normative outline for the `MakeConnection` assertion is:

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

480 The following describes the content model of the `MCSupported` element.

481 `/wsmc:MCSupported`

482 A policy assertion that specifies that the MakeConnection protocol is required for messages sent
483 from this endpoint.

484 /wsmc:MCSupported/{any}

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

487 /wsmc:MCSupported/@{any}

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

490 4 Faults

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

493 <http://docs.oasis-open.org/ws-rx/wsmc/200702/fault>

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

496 The definitions of faults use the following properties:

497 [Code] The fault code.

498 [Subcode] The fault subcode.

499 [Reason] The English language reason element.

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

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

SOAP Version	Sender	Receiver
SOAP 1.1	S11:Client	S11:Server
SOAP 1.2	S:Sender	S:Receiver

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

```
506 <S:Envelope>  
507 <S:Header>  
508 <wsa:Action>  
509 http://docs.oasis-open.org/ws-rx/wsmc/200702/fault  
510 </wsa:Action>  
511 <!-- Headers elided for brevity. -->  
512 </S:Header>  
513 <S:Body>  
514 <S:Fault>  
515 <S:Code>  
516 <S:Value> [Code] </S:Value>  
517 <S:Subcode>  
518 <S:Value> [Subcode] </S:Value>  
519 </S:Subcode>  
520 </S:Code>  
521 <S:Reason>  
522 <S:Text xml:lang="en"> [Reason] </S:Text>  
523 </S:Reason>  
524 <S:Detail>  
525 [Detail]  
526 ...  
527 </S:Detail>  
528 </S:Fault>  
529 </S:Body>  
530 </S:Envelope>
```

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

```

533 <S11:Envelope>
534 <S11:Body>
535 <S11:Fault>
536 <faultcode> [Subcode] </faultcode>
537 <faultstring> [Reason] </faultstring>
538 </S11:Fault>
539 </S11:Body>
540 </S11:Envelope>

```

541 4.1 Unsupported Selection

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

543 Properties:

544 [Code] Receiver

545 [Subcode] wsmc:UnsupportedSelection

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

548 [Detail]

```

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

```

Generated by	Condition	Action Upon Generation	Action Upon Receipt
MakeConnection receiver	In response to a MakeConnection message containing a selection criteria in the extensibility section of the message that is not supported	Unspecified.	Unspecified.

550 4.2 Missing Selection

551 The MakeConnection element did not contain any selection criteria.

552 Properties:

553 [Code] Receiver

554 [Subcode] wsmc:MissingSelection

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

556 [Detail]

Generated by	Condition	Action Upon Generation	Action Upon Receipt
--------------	-----------	------------------------	---------------------

Generated by	Condition	Action Upon Generation	Action Upon Receipt
MakeConnection receiver	In response to a MakeConnection message that does not contain any selection criteria	Unspecified.	Unspecified.

557 5 Security Considerations

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

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

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

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

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

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

596 Service endpoints SHOULD scope its searching of messages to those that were processed under the
597 same security context as the requesting `MakeConnection` message.

598 Appendix A. Schema

599 The normative schema that is defined for WS-MakeConnection using [XML-Schema Part1] and [XML-
600 Schema Part2] is located at:

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

602 The following copy is provided for reference.

```
603 <?xml version="1.0" encoding="UTF-8"?>
604 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
605      OASIS trademark, IPR and other policies apply. -->
606 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
607   xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsmc="http://docs.oasis-
608   open.org/ws-rx/wsmc/200702" targetNamespace="http://docs.oasis-open.org/ws-
609   rx/wsmc/200702" elementFormDefault="qualified"
610   attributeFormDefault="unqualified">
611   <xs:import namespace="http://www.w3.org/2005/08/addressing"
612   schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
613   <!-- Protocol Elements -->
614   <xs:complexType name="MessagePendingType">
615     <xs:sequence>
616       <xs:any namespace="##other" processContents="lax" minOccurs="0"
617   maxOccurs="unbounded"/>
618     </xs:sequence>
619     <xs:attribute name="pending" type="xs:boolean"/>
620     <xs:anyAttribute namespace="##other" processContents="lax"/>
621   </xs:complexType>
622   <xs:element name="MessagePending" type="wsmc:MessagePendingType"/>
623   <xs:element name="Address">
624     <xs:complexType>
625       <xs:simpleContent>
626         <xs:extension base="xs:anyURI">
627           <xs:anyAttribute namespace="##other" processContents="lax"/>
628         </xs:extension>
629       </xs:simpleContent>
630     </xs:complexType>
631   </xs:element>
632   <xs:complexType name="MakeConnectionType">
633     <xs:sequence>
634       <xs:element ref="wsmc:Address" minOccurs="0" maxOccurs="1"/>
635       <xs:any namespace="##other" processContents="lax" minOccurs="0"
636   maxOccurs="unbounded"/>
637     </xs:sequence>
638     <xs:anyAttribute namespace="##other" processContents="lax"/>
639   </xs:complexType>
640   <xs:element name="MakeConnection" type="wsmc:MakeConnectionType"/>
641   <xs:element name="UnsupportedSelection">
642     <xs:simpleType>
643       <xs:restriction base="xs:QName"/>
644     </xs:simpleType>
645   </xs:element>
646 </xs:schema>
```

647 Appendix B. WSDL

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

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

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

654 <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-wsdl-200702e1.wsdl>

655 The following non-normative copy is provided for reference.

```
656 <?xml version="1.0" encoding="utf-8"?>
657 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
658 OASIS trademark, IPR and other policies apply. -->
659 <wSDL:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
660 xmlns:xs="http://www.w3.org/2001/XMLSchema"
661 xmlns:wsa="http://www.w3.org/2005/08/addressing"
662 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
663 xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
664 xmlns:tns="http://docs.oasis-open.org/ws-rx/wsmc/200702/wsdl"
665 targetNamespace="http://docs.oasis-open.org/ws-rx/wsmc/200702/wsdl">
666
667   <wSDL:types>
668     <xs:schema
669       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsmc/200702"
670       schemaLocation="http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-schema-
671       200702.xsd"/>
672     </xs:schema>
673   </wSDL:types>
674
675   <wSDL:message name="MakeConnection">
676     <wSDL:part name="makeConnection" element="wsmc:MakeConnection"/>
677   </wSDL:message>
678
679   <wSDL:portType name="MCAbstractPortType">
680     <wSDL:operation name="MakeConnection">
681       <wSDL:input message="tns:MakeConnection" wsam:Action="http://docs.oasis-
682       open.org/ws-rx/wsmc/200702/MakeConnection"/>
683       <!-- As described in the WS-MakeConnection specification, the
684       MakeConnection operation establishes a connection. If a matching
685       message is available then the back-channel of the connection will
686       be used to carry the message. In SOAP terms the returned message
687       is not a response, so there is no WSDL output message. -->
688     </wSDL:operation>
689   </wSDL:portType>
690
691 </wSDL:definitions>
```

692 Appendix C. Message Examples

693 Appendix C.1 Example use of MakeConnection

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

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

```
702 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"  
703 xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"  
704 xmlns:wsrmp="http://docs.oasis-open.org/ws-rx/wsrmp/200702"  
705 xmlns:wsa="http://www.w3.org/2005/08/addressing">  
706   <S:Header>  
707     <wsa:To> http://example.org/subscriptionService </wsa:To>  
708     <wsa:MessageID> http://client456.org/id-a6d8-a7c2eb546813</wsa:MessageID>  
709     <wsa:ReplyTo>  
710       <wsa:To> http://client456.org/response </wsa:To>  
711     </wsa:ReplyTo>  
712   </S:Header>  
713   <S:Body>  
714     <sub:Subscribe xmlns:sub="http://example.org/subscriptionService">  
715       <!-- subscription service specific data -->  
716       <targetEPR>  
717         <wsa:Address>http://docs.oasis-open.org/ws-  
718 rx/wsrmp/200702/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:Address>  
719         <wsa:Metadata>  
720           <wsp:Policy wsu:Id="MyPolicy">  
721             <wsrmp:RMAssertion/>  
722           </wsp:Policy>  
723         </wsa:Metadata>  
724       </targetEPR>  
725     </sub:Subscribe>  
726   </S:Body>  
727 </S:Envelope>
```

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

733

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

```
735 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"  
736 xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"  
737 xmlns:wsa="http://www.w3.org/2005/08/addressing">  
738   <S:Header>  
739     <wsa:Action>http://docs.oasis-open.org/ws-  
740 rx/wsmc/200702/MakeConnection</wsa:Action>  
741     <wsa:To> http://example.org/subscriptionService </wsa:To>
```

```

742     </S:Header>
743     <S:Body>
744         <wsmc:MakeConnection>
745             <wsmc:Address>http://docs.oasis-open.org/ws-
746 rx/wsmc/200702/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsmc:Address>
747         </wsmc:MakeConnection>
748     </S:Body>
749 </S:Envelope>

```

750 **Step 3** – If there are messages waiting to be delivered then a message will be returned back to the event
751 consumer. However, because WS-RM is being used to deliver the messages, the first message returned
752 is a CreateSequence:

```

753 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
754 xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
755 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
756 xmlns:wsa="http://www.w3.org/2005/08/addressing">
757     <S:Header>
758         <wsa:Action>http://docs.oasis-open.org/ws-
759 rx/wsmr/200702/CreateSequence</wsa:Action>
760         <wsa:To>http://docs.oasis-open.org/ws-rx/wsmr/200702/anonymous?id=550e8400-
761 e29b-11d4-a716-446655440000</wsa:To>
762         <wsa:ReplyTo> http://example.org/subscriptionService </wsa:ReplyTo>
763         <wsa:MessageID> http://example.org/id-123-456 </wsa:MessageID>
764         <wsmc:MessagePending pending="true"/>
765     </S:Header>
766     <S:Body>
767         <wsmr:CreateSequence>
768             <wsrm:AcksTo>
769                 <wsa:Address> http://example.org/subscriptionService </wsa:Address>
770             </wsrm:AcksTo>
771         </wsmr:CreateSequence>
772     </S:Body>
773 </S:Envelope>

```

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

778

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

```

781 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
782 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
783 xmlns:wsa="http://www.w3.org/2005/08/addressing">
784     <S:Header>
785         <wsa:Action>http://docs.oasis-open.org/ws-
786 rx/wsmr/200702/CreateSequenceResponse</wsa:Action>
787         <wsa:To> http://example.org/subscriptionService </wsa:To>
788         <wsa:RelatesTo> http://example.org/id-123-456 </wsa:RelatesTo>
789     </S:Header>
790     <S:Body>
791         <wsmr:CreateSequenceResponse>
792             <wsrm:Identifier> http://example.org/rmid-456 </wsrm:Identifier>
793         </wsmr:CreateSequenceResponse>
794     </S:Body>
795 </S:Envelope>

```

796 Note, this message is carried on an HTTP request directed to the `wsa:ReplyTo` EPR, and the HTTP
797 response will be an HTTP 202.

798

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

```
800 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"  
801 xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"  
802 xmlns:wsa="http://www.w3.org/2005/08/addressing">  
803   <S:Header>  
804     <wsa:Action>http://docs.oasis-open.org/ws-  
805 rx/wsmc/200702/MakeConnection</wsa:Action>  
806     <wsa:To> http://example.org/subscriptionService </wsa:To>  
807   </S:Header>  
808   <S:Body>  
809     <wsmc:MakeConnection>  
810       <wsmc:Address>http://docs.oasis-open.org/ws-  
811 rx/wsmc/200702/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsmc:Address>  
812     </wsmc:MakeConnection>  
813   </S:Body>  
814 </S:Envelope>
```

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

816

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

```
818 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"  
819 xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"  
820 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"  
821 xmlns:wsa="http://www.w3.org/2005/08/addressing">  
822   <S:Header>  
823     <wsa:Action> http://example.org/eventType1</wsa:Action>  
824     <wsa:To>http://docs.oasis-open.org/ws-rx/wsmr/200702/anonymous?id=550e8400-  
825 e29b-11d4-a716-446655440000</wsa:To>  
826     <wsmr:Sequence>  
827       <wsmr:Identifier> http://example.org/rmid-456 </wsmr:Identifier>  
828     </wsmr:Sequence>  
829     <wsmc:MessagePending pending="true"/>  
830   </S:Header>  
831   <S:Body>  
832     <!-- event specific data -->  
833   </S:Body>  
834 </S:Envelope>
```

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

838

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

```
841 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"  
842 xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"  
843 xmlns:wsa="http://www.w3.org/2005/08/addressing">  
844   <S:Header>  
845     <wsa:Action> http://docs.oasis-open.org/ws-rx/wsmc/200702/MakeConnection  
846   </wsa:Action>  
847     <wsa:To> http://example.org/subscriptionService </wsa:To>  
848   </S:Header>
```

```
849 <S:Body>
850 <wsmc:MakeConnection>
851 <wsmc:Address>http://docs.oasis-open.org/ws-
852 rx/wsmc/200702/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsmc:Address>
853 </wsmc:MakeConnection>
854 </S:Body>
855 </S:Envelope>
```

856 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
857 `MakeConnection` can be any message destined to the specified Endpoint. This allows the event
858 producer to send not only application messages (events) but RM protocol messages (e.g.
859 `CloseSequence`, `TerminateSequence` or even additional `CreateSequence` messages) as needed.

860

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

864 Appendix D. Acknowledgments

865 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 Bjarlestam, 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 Mischkin, 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 Dournaee, 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 Raeppe, SAP
913	Paul Fremantle, WSO2	943	Eric Rajkovic, Oracle
914	Robert Freund, Hitachi	944	Stefan Rossmann, 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 Videlov, SAP
919	Ondrej Hrebicek, Microsoft	949	Claus von Riegen, SAP
920	Kazunori Iwasa, Fujitsu	950	Pete Wenzel, Sun
921	Chamikara Jayalath, 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