



1 Web Services Make Connection 2 (WS-MakeConnection) 1.0

3 Committee Draft 01

4 1 February 2007

5 Specification URIs:

6 This Version:

7 <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-cd-01.pdf>

8 <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-cd-01.html>

9 Latest Version:

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

11 Latest Approved Version:

12 <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-cd-01.pdf>

13 <http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-cd-01.html>

14 Technical Committee:

15 OASIS Web Services Reliable Exchange (WS-RX) TC

16 Chairs:

17 Paul Fremantle <paul@wso2.com>

18 Sanjay Patil <sanjay.patil@sap.com>

19 Editors:

20 Doug Davis, IBM <dug@us.ibm.com>

21 Anish Karmarkar, Oracle <Anish.Karmarkar@oracle.com>

22 Gilbert Pilz, BEA <gpilz@bea.com>

23 Steve Winkler, SAP <steve.winkler@sap.com>

24 Ümit Yalçınalp, SAP <umit.yalcinalp@sap.com>

25 Declared XML Namespaces:

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

27 Abstract:

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

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

36 By using the XML [[XML](#)], SOAP [[SOAP 1.1](#)], [[SOAP 1.2](#)] and WSDL [[WSDL 1.1](#)] extensibility model,
37 SOAP-based and WSDL-based specifications are designed to be composed with each other to define a

38 rich Web services environment. As such, WS-MakeConnection by itself does not define all the
39 features required for a complete messaging solution. WS-MakeConnection is a building block that
40 is used in conjunction with other specifications and application-specific protocols to accommodate
41 a wide variety of requirements and scenarios related to the operation of distributed Web services.

42 **Status:**

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

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

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

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

56 Notices

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

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

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

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

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

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

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

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

96 The name "OASIS" is a trademark of [OASIS](http://www.oasis-open.org), the owner and developer of this specification, and should be
97 used only to refer to the organization and its official outputs. OASIS welcomes reference to, and
98 implementation and use of, specifications, while reserving the right to enforce its marks against
99 misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

100	Table of Contents	
101	1 Introduction.....	6
102	1.1 Terminology.....	6
103	1.2 Namespace.....	7
104	1.3 Conformance.....	7
105	2 MakeConnection Model.....	8
106	2.1 Glossary.....	9
107	2.2 Protocol Preconditions.....	9
108	2.3 Example Message Exchange.....	9
109	3 MakeConnection.....	11
110	3.1 MakeConnection Anonymous URI.....	11
111	3.2 MakeConnection Message.....	11
112	3.3 MessagePending.....	13
113	3.4 MakeConnection Policy Assertion.....	13
114	4 Faults.....	15
115	4.1 Unsupported Selection.....	16
116	4.2 Missing Selection.....	16
117	5 Security Considerations.....	18
118	6 References.....	19
119	6.1 Normative.....	19
120	6.2 Non-Normative.....	20
121	Appendix A. Schema.....	22
122	Appendix B. WSDL.....	23
123	Appendix C. Message Examples.....	24
124	Appendix C.1 Example use of MakeConnection.....	24
125	Appendix D. Acknowledgments.....	28
126	Appendix E. Revision History.....	29

127 **1 Introduction**

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

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

137 **1.1 Terminology**

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

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

- 142 • The syntax appears as an XML instance, but values in italics indicate data types instead of values.
- 143 • Characters are appended to elements and attributes to indicate cardinality:
 - 144 ○ "?" (0 or 1)
 - 145 ○ "*" (0 or more)
 - 146 ○ "+" (1 or more)
- 147 • The character "|" is used to indicate a choice between alternatives.
- 148 • The characters "[" and "]" are used to indicate that contained items are to be treated as a group
149 with respect to cardinality or choice.
- 150 • An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content
151 specified in this document. Additional children elements and/or attributes MAY be added at the
152 indicated extension points but they MUST NOT contradict the semantics of the parent and/or
153 owner, respectively. If an extension is not recognized it SHOULD be ignored.
- 154 • XML namespace prefixes (See Section [1.2](#)) are used to indicate the namespace of the element
155 being defined.

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

- 159 • An element extensibility point is referred to using {any} in place of the element name. This
160 indicates that any element name can be used, from any namespace other than the `wsmc :`
161 namespace.
- 162 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This
163 indicates that any attribute name can be used, from any namespace other than the `wsmc :`
164 namespace.

165 1.2 Namespace

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

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

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

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

172 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/2007/02/addressing/metadata
xs	http://www.w3.org/2001/XMLSchema

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

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

176 1.3 Conformance

177 An implementation is not conformant with this specification if it fails to satisfy one or more of the MUST or
178 REQUIRED level requirements defined herein. A SOAP Node MUST NOT use the XML namespace
179 identifier for this specification (listed in Section 1.2) within SOAP Envelopes unless it is conformant with
180 this specification.

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

183 2 MakeConnection Model

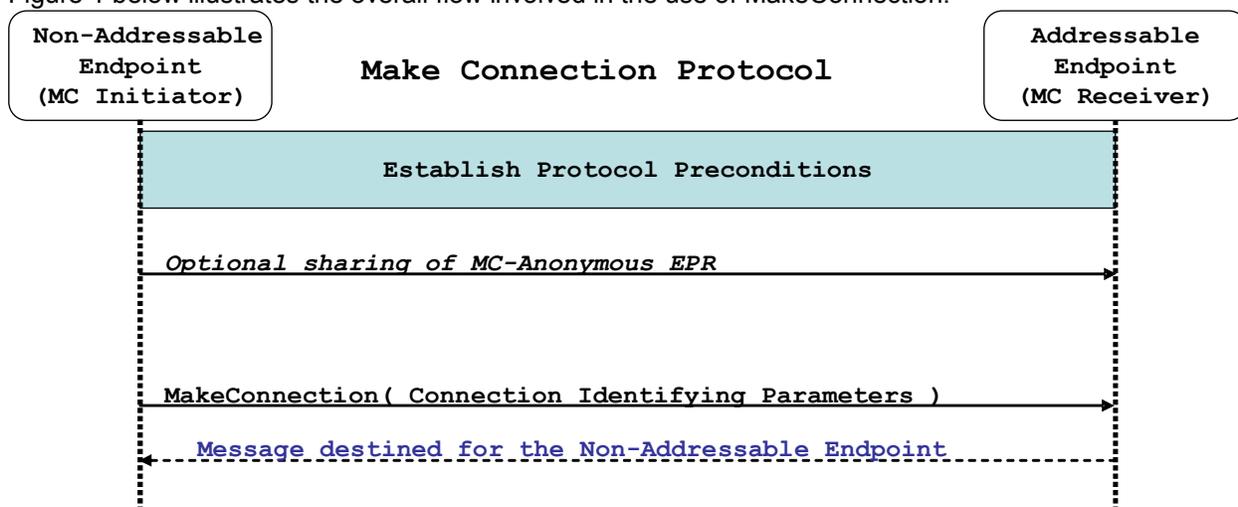
184 The WS-Addressing [WS-Addressing] specification defines the anonymous URI to identify non-
185 addressable endpoints and to indicate a protocol-specific back-channel is to be used for any messages
186 destined for that endpoint. For example, when used in the WS-Addressing ReplyTo EPR, the use of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

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

199 The MakeConnection model consists of a two key aspects:

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

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



206 Figure 1 – Make Connection Model

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

210 2.1 Glossary

211 The following definitions are used throughout this specification:

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

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

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

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

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

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

224 2.2 Protocol Preconditions

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

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

232 2.3 Example Message Exchange

233 Figure 2 illustrates a message exchange in which the response message is delivered using
 234 MakeConnection.



235 Figure 2: Example WS-MakeConnection Message Exchange

- 236 1. The protocol preconditions are established. These include policy exchange, endpoint resolution,
237 and establishing trust.
- 238 2. The client (MC Initiator) sends a GetQuote request message to the service (MC Receiver). The
239 WS-Addressing `wsa:ReplyTo` EPR uses the MakeConnection Anonymous URI Template –
240 indicating that if the GetQuoteResponse message is not sent back on this connection's back-
241 channel, then the client will use MakeConnection to retrieve it.
- 242 3. The service receives the request message and decides to close the connection by sending back an
243 empty response (in the HTTP case an HTTP 202 Accept is sent).
- 244 4. The client sends a MakeConnection message to the service. Within the MakeConnection element is
245 the `wsmc:Address` element containing the same MakeConnection Anonymous URI used in step 2.
- 246 5. The service has not completed executing the GetQuote operation and decides to close the
247 connection by sending back an empty response (in the HTTP case an HTTP 202 Accept) indicating
248 that no messages destined for this MC Initiator are available at this time.
- 249 6. The client sends a second MakeConnection message to the service. Within the MakeConnection
250 element is the `wsmc:Address` element containing the same MakeConnection Anonymous URI
251 used in step 2.
- 252 7. The service uses this new connection to transmit the GetQuoteResponse message.

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

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

269 **3 MakeConnection**

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

272 **3.1 MakeConnection Anonymous URI**

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

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

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

287 **3.2 MakeConnection Message**

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

293 The following exemplar defines the `MakeConnection` syntax:

```
294 <wsmc:MakeConnection ...>  
295   <wsa:Address ...> xs:anyURI </wsa:Address> ?  
296   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier> ?  
297   ...  
298 </wsmc:MakeConnection>
```

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

300 `/wsmc:MakeConnection`

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

305 `/wsmc:MakeConnection/wsmc:Address`

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

310 may in fact be functionally equivalent. Examples include URI references which differ only in case, or
311 which are in external entities which have different effective base URIs.

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

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

315 `/wsmc:MakeConnection/wsrn:Identifier`

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

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

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

323 `/wsmc:MakeConnection/{any}`

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

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

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

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

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

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

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

- 347 ● The `MakeConnection` message is logically part of a one-way operation; there is no reply
348 message to the `MakeConnection` itself, and any response flowing on the transport back-channel
349 is a pending message.
- 350 ● Since there is no reply message to `MakeConnection`, the WS-Addressing specific rules in
351 section 3.4 "Formulating a Reply Message" are not used. Therefore, the value of any

352 `wsa:ReplyTo` element in the `MakeConnection` message has no effective impact since the WS-
353 Addressing [`reply endpoint`] property that is set by the presence of `wsa:ReplyTo` is not
354 used.

- 355 ● In the absence of any pending message, there will be no message transmitted on the transport
356 back-channel. E.g. in the HTTP case just an `HTTP 202 Accepted` will be returned without any
357 SOAP envelope in the HTTP response message.
- 358 ● When there is a message pending, it is sent on the transport back-channel, using the connection
359 that has been initiated by the `MakeConnection` request.

360 **3.3 MessagePending**

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

365 The following exemplar defines the `MessagePending` syntax:

```
366 <wsmc:MessagePending pending="xs:boolean" ...>  
367   ...  
368 </wsmc:MessagePending>
```

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

370 `/wsmc:MessagePending`

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

372 `/wsmc:MessagePending@pending`

373 This attribute, when set to "true", indicates that there is at least one message waiting to be retrieved.

374 When this attribute is set to "false" it indicates there are currently no messages waiting to be retrieved.

375 `/wsmc:MessagePending/{any}`

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

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

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

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

383 **3.4 MakeConnection Policy Assertion**

384 The `MakeConnection` policy assertion indicates that the `MakeConnection` protocol (operation and the use
385 of the `MakeConnection` URI template in `EndpointReferences`) is supported. This assertion has `Endpoint`
386 `Policy Subject [WS-PolicyAttachment]`.

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

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

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

390 /wsmc:MCSupported

391 A policy assertion that specifies that the MakeConnection protocol is supported.

392 /wsmc:MCSupported/{any}

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

395 /wsmc:MCSupported/@{any}

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

398 Because this policy assertion expresses a capability of a receiver (rather than a requirement on the
399 sender), care should be taken to ensure that it is decorated with the appropriate WS-Policy artifacts to
400 indicate that use, support and understanding, of this assertion is optional to the sender.

401 4 Faults

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

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

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

407 The definitions of faults use the following properties:

408 [Code] The fault code.

409 [Subcode] The fault subcode.

410 [Reason] The English language reason element.

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

414 Entities that generate WS-MakeConnection faults MUST set the [Code] property to either "Sender" or
415 "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

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

```
417 <S:Envelope>
418   <S:Header>
419     <wsa:Action>
420       http://docs.oasis-open.org/ws-rx/wsmc/200702/fault
421     </wsa:Action>
422     <!-- Headers elided for brevity. -->
423   </S:Header>
424   <S:Body>
425     <S:Fault>
426       <S:Code>
427         <S:Value> [Code] </S:Value>
428         <S:Subcode>
429           <S:Value> [Subcode] </S:Value>
430         </S:Subcode>
431       </S:Code>
432       <S:Reason>
433         <S:Text xml:lang="en"> [Reason] </S:Text>
434       </S:Reason>
435       <S:Detail>
436         [Detail]
437       ...
438     </S:Detail>
439   </S:Fault>
440 </S:Body>
441 </S:Envelope>
```

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

```

444 <S11:Envelope>
445   <S11:Body>
446     <S11:Fault>
447       <faultcode> [Subcode] </faultcode>
448       <faultstring> [Reason] </faultstring>
449     </S11:Fault>
450   </S11:Body>
451 </S11:Envelope>

```

452 4.1 Unsupported Selection

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

454 Properties:

455 [Code] Receiver

456 [Subcode] wsmc:UnsupportedSelection

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

459 [Detail]

```

460 <wsmc:UnsupportedElement> xs:QName </wsmc:UnsupportedElement>+

```

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.

461 4.2 Missing Selection

462 The MakeConnection element did not contain any selection criteria.

463 Properties:

464 [Code] Receiver

465 [Subcode] wsmc:MissingSelection

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

467 [Detail]

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.

468 **5 Security Considerations**

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

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

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

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

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

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

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

509 **6 References**

510 **6.1 Normative**

511 **[KEYWORDS]**

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

514 <http://www.ietf.org/rfc/rfc2119.txt>

515 **[WS-RM]**

516 OASIS WS-RX Technical Committee Draft, "Web Services Reliable Messaging (WS-ReliableMessaging),"
517 August 2005.

518 <http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-cd-05.pdf>

519 **[WS-RM Policy]**

520 OASIS WS-RX Technical Committee Draft, "Web Services ReliableMessaging Policy Assertion(WS-RM
521 Policy)" February 2007

522 <http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-cd-05.pdf>

523 **[SOAP 1.1]**

524 W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.

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

526 **[SOAP 1.2]**

527 W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework" June 2003.

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

529 **[URI]**

530 T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 3986,
531 MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005.

532 <http://ietf.org/rfc/rfc3986>

533 **[UUID]**

534 P. Leach, M. Mealling, R. Salz, "A Universally Unique Identifier (UUID) URN Namespace," RFC 4122,
535 Microsoft, Refactored Networks - LLC, DataPower Technology Inc, July 2005

536 <http://www.ietf.org/rfc/rfc4122.txt>

537 **[XML]**

538 W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", September 2006.

539 <http://www.w3.org/TR/REC-xml/>

540 **[XML-ns]**

541 W3C Recommendation, "Namespaces in XML," 14 January 1999.

542 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
543 **[XML-Schema Part1]**
544 W3C Recommendation, "XML Schema Part 1: Structures," October 2004.
545 <http://www.w3.org/TR/xmlschema-1/>
546 **[XML-Schema Part2]**
547 W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.
548 <http://www.w3.org/TR/xmlschema-2/>
549 **[XPath 1.0]**
550 W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16 November 1999.
551 <http://www.w3.org/TR/xpath>
552 **[WSDL 1.1]**
553 W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.
554 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
555 **[WS-Addressing]**
556 W3C Recommendation, "Web Services Addressing 1.0 - Core", May 2006.
557 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
558 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding", May 2006.
559 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

560 **6.2 Non-Normative**

561 **[BSP 1.0]**
562 WS-I Working Group Draft. "Basic Security Profile Version 1.0," August 2006
563 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>
564 **[RDDL 2.0]**
565 Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language (RDDL) 2.0," January 2004
566 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>
567 **[RFC 2617]**
568 J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L. Stewart, "HTTP
569 Authentication: Basic and Digest Access Authentication," June 1999.
570 <http://www.ietf.org/rfc/rfc2617.txt>
571 **[RFC 4346]**
572 T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," April 2006.
573 <http://www.ietf.org/rfc/rfc4346.txt>
574 **[WS-Policy]**
575 W3C Member Submission, "Web Services Policy Framework (WS-Policy)," April 2006.

576 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>
577 **[WS-PolicyAttachment]**
578 W3C Member Submission, "[Web Services Policy Attachment \(WS-PolicyAttachment\)](#)," April 2006.
579 [http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)
580 [20060425/](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)
581 **[WS-Security]**
582 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "[OASIS Web Services Security: SOAP Message Security 1.0 \(WS-Security 2004\)](#)", OASIS Standard 200401, March 2004.
583 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>
584 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "[OASIS Web Services Security: SOAP Message Security 1.1 \(WS-Security 2004\)](#)", OASIS Standard 200602, February 2006.
585 <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
586 **[RTTM]**
587 V. Jacobson, R. Braden, D. Borman, "[TCP Extensions for High Performance](#)", RFC 1323, May
588 1992.
589 <http://www.rfc-editor.org/rfc/rfc1323.txt>
590
591 **[SecurityPolicy]**
592 G. Della-Libra, et. al. "[Web Services Security Policy Language \(WS-SecurityPolicy\)](#)", July 2005
593 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>
594 **[SecureConversation]**
595 S. Anderson, et al, "[Web Services Secure Conversation Language \(WS-SecureConversation\)](#)," February
596 2005.
597 <http://schemas.xmlsoap.org/ws/2004/04/sc/>
598 **[Trust]**
599 S. Anderson, et al, "[Web Services Trust Language \(WS-Trust\)](#)," February 2005.
600 <http://schemas.xmlsoap.org/ws/2005/02/trust>
601

602 Appendix A. Schema

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

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

606 The following copy is provided for reference.

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

651 Appendix B. WSDL

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

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

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

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

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

```
660 <?xml version="1.0" encoding="utf-8"?>
661 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
662 OASIS trademark, IPR and other policies apply. -->
663 <wSDL:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
664 xmlns:xs="http://www.w3.org/2001/XMLSchema"
665 xmlns:wsa="http://www.w3.org/2005/08/addressing"
666 xmlns:wsam="http://www.w3.org/2007/02/addressing/metadata"
667 xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
668 xmlns:tns="http://docs.oasis-open.org/ws-rx/wsmc/200702/wsd1"
669 targetNamespace="http://docs.oasis-open.org/ws-rx/wsmc/200702/wsd1">
670
671   <wSDL:types>
672     <xs:schema>
673       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsmc/200702"
674       schemaLocation="http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.1-schema-
675       200702.xsd"/>
676     </xs:schema>
677   </wSDL:types>
678
679   <wSDL:message name="MakeConnection">
680     <wSDL:part name="makeConnection" element="wsmc:MakeConnection"/>
681   </wSDL:message>
682
683   <wSDL:portType name="MCAbstractPortType">
684     <wSDL:operation name="MakeConnection">
685       <wSDL:input message="tns:MakeConnection" wsam:Action="http://docs.oasis-
686       open.org/ws-rx/wsmc/200702/MakeConnection"/>
687       <!-- As described in the WS-MakeConnection specification, the
688       MakeConnection operation establishes a connection. If a matching
689       message is available then the back-channel of the connection will
690       be used to carry the message. In SOAP terms the returned message
691       is not a response, so there is no WSDL output message. -->
692     </wSDL:operation>
693   </wSDL:portType>
694 </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
701 and the WS-RM Policy Assertion 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://exaaple.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
732 the RM must be used when notifications related to this subscription are sent.

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

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

```

740     <wsa:To> http://example.org/subscriptionService </wsa:To>
741   </S:Header>
742   <S:Body>
743     <wsmc:MakeConnection>
744       <wsmc:Address>http://docs.oasis-open.org/ws-
745 rx/wsmc/200702/anonymous?id=550e8400-e29b-11d4-a716-
746 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-
761 rx/wsmr/200702/anonymous?id=550e8400-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       <wsmr:AcksTo>
769         <wsa:Address> http://example.org/subscriptionService </wsa:Address>
770       </wsmr: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
775 Destination of those messages, nothing new is introduced by the use of the `MakeConnection`, the use
776 of RM protocol is the same as the case where the event consumer is addressable. Note the message
777 contains a `wsmc:MessagePending` header indicating that additional message are waiting to be
778 delivered.

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       <wsmr:Identifier> http://example.org/rmid-456 </wsmr:Identifier>

```

```
793     </wsrm: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 **Step 5** – The event consumer checks for another message pending:

```
799 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
800 xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
801 xmlns:wsa="http://www.w3.org/2005/08/addressing">
802   <S:Header>
803     <wsa:Action>http://docs.oasis-open.org/ws-
804 rx/wsmc/200702/MakeConnection</wsa:Action>
805     <wsa:To> http://example.org/subscriptionService </wsa:To>
806   </S:Header>
807   <S:Body>
808     <wsmc:MakeConnection>
809       <wsmc:Address>http://docs.oasis-open.org/ws-
810 rx/wsmc/200702/anonymous?id=550e8400-e29b-11d4-a716-
811 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 **Step 6** – Since there is a message pending for this destination then it is returned on the HTTP response:

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

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

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

```
839 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
840 xmlns:wsmc="http://docs.oasis-open.org/ws-rx/wsmc/200702"
841 xmlns:wsa="http://www.w3.org/2005/08/addressing">
```

```
842 <S:Header>
843 <wsa:Action>http://docs.oasis-open.org/ws-
844 rx/wsmc/200702/MakeConnection</wsa:Action>
845 <wsa:To> http://example.org/subscriptionService </wsa:To>
846 </S:Header>
847 <S:Body>
848 <wsmc:MakeConnection>
849 <wsmc:Address>http://docs.oasis-open.org/ws-
850 rx/wsmc/200702/anonymous?id=550e8400-e29b-11d4-a716-
851 446655440000</wsmc:Address>
852 </wsmc:MakeConnection>
853 </S:Body>
854 </S:Envelope>
```

855 Notice this is the same message as the one sent in steps 2 and 5. As in steps 3 and 6, the response to
856 the `MakeConnection` can be any message destined to the specified Endpoint. This allows the event
857 producer to send not only application messages (events) but RM protocol messages (e.g.
858 `CloseSequence`, `TerminateSequence` or even additional `CreateSequences`) as needed.

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

862 **Appendix D. Acknowledgments**

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

864 Keith Ballinger(Microsoft), Stefan Batres(Microsoft), Rebecca Bergersen(Iona), Allen
865 Brown(Microsoft), Kyle Brown(IBM), Michael Conner(IBM), George Copeland(Microsoft),
866 Francisco Curbera(IBM), Paul Fremantle(IBM), Steve Graham(IBM), Pat Helland(Microsoft), Rick
867 Hill(Microsoft), Scott Hinkelman(IBM), Tim Holloway(IBM), Efim Hudis(Microsoft), David
868 Ingham(Microsoft), Gopal Kakivaya(Microsoft), Johannes Klein(Microsoft), Frank Leymann(IBM),
869 Martin Nally(IBM), Peter Niblett(IBM), Jeffrey Schlimmer(Microsoft), James Snell(IBM), Keith
870 Stobie(Microsoft), Satish Thatte(Microsoft), Stephen Todd(IBM), Sanjiva Weerawarana(IBM),
871 Roger Wolter(Microsoft).

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

873 Abbie Barbir(Nortel), Charlton Barreto(Adobe), Stefan Batres(Microsoft), Hamid Ben
874 Malek(Fujitsu), Andreas Bjarlestam(Ericsson), Toufic Boubez(Layer 7), Doug Bunting(Sun), Lloyd
875 Burch(Novell), Steve Carter(Novell), Martin Chapman(Oracle), Dave Chappell(Sonic), Paul
876 Cotton(Microsoft), Glen Daniels(Sonic), Doug Davis(IBM), Blake Dournaee(Intel), Jacques
877 Durand(Fujitsu), Colleen Evans(Microsoft), Christopher Ferris(IBM), Paul Fremantle(WSO2),
878 Robert Freund(Hitachi), Peter Furniss(Erebor), Marc Goodner(Microsoft), Alastair
879 Green(Choreology), Mike Grogan(Sun), Ondrej Hrebicek(Microsoft), Kazunori Iwasa(Fujitsu),
880 Chamikara Jayalath(WSO2), Lei Jin(BEA), Ian Jones(BTplc), Anish Karmarkar(Oracle), Paul
881 Knight(Nortel), Dan Leshchiner(Tibco), Mark Little(JBoss), Lily Liu(webMethods), Matt
882 Lovett(IBM), Ashok Malhotra(Oracle), Jonathan Marsh(Microsoft), Daniel Millwood(IBM), Jeff
883 Mischkinsky(Oracle), Nilo Mitra(Ericsson), Peter Niblett(IBM), Duane Nickull(Adobe), Eisaku
884 Nishiyama(Hitachi), Dave Orchard(BEA), Chouthri Palanisamy(NEC), Sanjay Patil(SAP), Gilbert
885 Pilz(BEA), Martin Raeppe(SAP), Eric Rajkovic(Oracle), Stefan Rossmannith(SAP), Tom
886 Rutt(Fujitsu), Rich Salz(IBM), Shivajee Samdarshi(Tibco), Vladimir Videlov(SAP), Claus von
887 Riegen(SAP), Pete Wenzel(Sun), Steve Winkler(SAP), Ümit Yalçinalp(SAP), Nobuyuki
888 Yamamoto(Hitachi).

889 **Appendix E. Revision History**

Rev	Date	By Whom	What
wd-01	2006-12-31	Doug Davis	Initial version created based on section 10(MakeConnection) in the WS-RM spec
wd-02	2007-01-31	Doug Davis	Lots of typos from MarcG Updated WD number and date
wd-02	2007-02-01	Doug Davis	PR015 and PR029 applied