
Web Services for Management (WS-Management) Specification

*Spécification des services Web pour le management
(WS-Management)*





COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2013

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 17963 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 38, *Distributed application platforms and services (DAPS)*.



Document Number: DSP0226

Date: 2012-08-28

Version: 1.1.1

Web Services for Management (WS- Management) Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: en-US

10 Copyright Notice

11 Copyright © 2006–2012 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

12 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
13 management and interoperability. Members and non-members may reproduce DMTF specifications and
14 documents, provided that correct attribution is given. As DMTF specifications may be revised from time to
15 time, the particular version and release date should always be noted.

16 Implementation of certain elements of this standard or proposed standard may be subject to third party
17 patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations
18 to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose,
19 or identify any or all such third party patent right, owners or claimants, nor for any incomplete or
20 inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to
21 any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize,
22 disclose, or identify any such third party patent rights, or for such party's reliance on the standard or
23 incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any
24 party implementing such standard, whether such implementation is foreseeable or not, nor to any patent
25 owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is
26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party
27 implementing the standard from any and all claims of infringement by a patent owner for such
28 implementations.

29 For information about patents held by third-parties which have notified the DMTF that, in their opinion,
30 such patent may relate to or impact implementations of DMTF standards, visit
31 <http://www.dmtf.org/about/policies/disclosures.php>.

32

CONTENTS

34	Foreword.....	7
35	1 Scope.....	10
36	2 Normative References.....	10
37	3 Terms and Definitions.....	12
38	4 Symbols and Abbreviated Terms.....	15
39	5 Addressing.....	16
40	5.1 Management Addressing.....	16
41	5.2 Versions of Addressing.....	25
42	5.3 Requirements for Compatibility.....	25
43	5.4 Use of Addressing in WS-Management.....	27
44	6 WS-Management Control Headers.....	44
45	6.1 wsman:OperationTimeout.....	44
46	6.2 wsman:MaxEnvelopeSize.....	45
47	6.3 wsman:Locale.....	46
48	6.4 wsman:OptionSet.....	47
49	6.5 wsman:RequestEPR.....	50
50	7 Resource Access.....	51
51	7.1 General.....	51
52	7.2 Addressing Uniformity.....	53
53	7.3 Get.....	54
54	7.4 Put.....	55
55	7.5 Delete.....	59
56	7.6 Create.....	61
57	7.7 Fragment-Level Access.....	64
58	7.8 Fragment-Level Get.....	66
59	7.9 Fragment-Level Put.....	67
60	7.10 Fragment-Level Delete.....	70
61	7.11 Fragment-Level Create.....	71
62	8 Enumeration of Datasets.....	73
63	8.1 General.....	73
64	8.2 Enumerate.....	75
65	8.3 Filter Interpretation.....	82
66	8.4 Pull.....	84
67	8.5 Release.....	88
68	8.6 Ad-Hoc Queries and Fragment-Level Enumerations.....	90
69	8.7 Enumeration of EPRs.....	90
70	8.8 Renew.....	92
71	8.9 GetStatus.....	94
72	8.10 EnumerationEnd.....	94
73	9 Custom Actions (Methods).....	95
74	10 Notifications (Eventing).....	96
75	10.1 General.....	96
76	10.2 Subscribe.....	97
77	10.3 GetStatus.....	117
78	10.4 Unsubscribe.....	118
79	10.5 Renew.....	119
80	10.6 SubscriptionEnd.....	120
81	10.7 Acknowledgement of Delivery.....	122
82	10.8 Refusal of Delivery.....	123
83	10.9 Dropped Events.....	124
84	10.10 Access Control.....	125

85	10.11 Implementation Considerations.....	126
86	10.12 Advertisement of Notifications.....	126
87	11 Metadata and Discovery	126
88	12 Security	129
89	12.1 General.....	129
90	12.2 Security Profiles	130
91	12.3 Security Considerations for Event Subscriptions	130
92	12.4 Including Credentials with a Subscription	131
93	12.5 Correlating Events with a Subscription	132
94	12.6 Transport-Level Authentication Failure	132
95	12.7 Security Implications of Third-Party Subscriptions.....	132
96	13 Transports and Message Encoding.....	133
97	13.1 SOAP.....	133
98	13.2 Lack of Response.....	134
99	13.3 Replay of Messages.....	134
100	13.4 Encoding Limits	134
101	13.5 Binary Attachments	135
102	13.6 Case-Sensitivity.....	135
103	14 Faults	136
104	14.1 Introduction	136
105	14.2 Fault Encoding	136
106	14.3 NotUnderstood Faults	137
107	14.4 Degenerate Faults	138
108	14.5 Fault Extensibility	138
109	14.6 Master Faults.....	139
110	ANNEX A (informative) Notational Conventions	160
111	A.1 XML Namespaces	160
112	ANNEX B (normative) Conformance	162
113	ANNEX C (normative) HTTP(S) Transport and Security Profile	163
114	C.1 General.....	163
115	C.2 HTTP(S) Binding	163
116	C.3 HTTP(S) Security Profiles	165
117	C.4 IPSec and HTTP	170
118	ANNEX D (informative) XPath Support	171
119	D.1 General.....	171
120	D.2 Level 1	172
121	D.3 Level 2.....	174
122	ANNEX E (normative) Selector Filter Dialect	177
123	ANNEX F (informative) Identify XML Schema.....	179
124	ANNEX G (informative) Resource Access Operations XML Schema and WSDL	182
125	ANNEX H (informative) Enumeration Operations XML Schema and WSDL	187
126	ANNEX I (informative) Notification OperationsXML Schema and WSDL	196
127	ANNEX J (informative) Addressing XML Schema.....	204
128	ANNEX K (informative) WS-Management XML Schema.....	207
129	ANNEX L (informative) Change Log.....	217
130		

131 **Figures**

132	Figure 1 – Message Information Header Blocks	20
-----	--	----

133

134 **Tables**

135	Table 1 – Relationship Type	21
136	Table 2 – Interoperability Requirements	25
137	Table 3 – WSA Versions in Exchanges	26
138	Table 4 – wsa:Action URI Descriptions	42
139	Table 5 – wsman:AccessDenied	139
140	Table 6 – wsa:ActionNotSupported	140
141	Table 7 – wsman:AlreadyExists	140
142	Table 8 – wsmen:CannotProcessFilter	141
143	Table 9 – wsman:CannotProcessFilter	141
144	Table 10 – wsman:Concurrency	142
145	Table 11 – wsme:DeliveryModeRequestedUnavailable	142
146	Table 12 – wsman:DeliveryRefused	143
147	Table 13 – wsa:DestinationUnreachable	143
148	Table 14 – wsman:EncodingLimit	144
149	Table 15 – wsa:EndpointUnavailable	145
150	Table 16 – wsman:EventDeliverToUnusable	145
151	Table 17 – wsme:EventSourceUnableToProcess	146
152	Table 18 – wsmen:FilterDialectRequestedUnavailable	146
153	Table 19 – wsme:FilteringNotSupported	146
154	Table 20 – wsmen:FilteringNotSupported	147
155	Table 21 – wsme:FilteringRequestedUnavailable	147
156	Table 22 – wsman:FragmentDialectNotSupported	148
157	Table 23 – wsman:InternalError	148
158	Table 24 – wsman:InvalidBookmark	149
159	Table 25 – wsmen:InvalidEnumerationContext	149
160	Table 26 – wsme:InvalidExpirationTime	150
161	Table 27 – wsmen:InvalidExpirationTime	150
162	Table 28 – wsme:InvalidMessage	151
163	Table 29 – wsa:InvalidMessageInformationHeader	151
164	Table 30 – wsman:InvalidOptions	152
165	Table 31 – wsman:InvalidParameter	152
166	Table 32 – wsmt:InvalidRepresentation	153
167	Table 33 – wsman:InvalidSelectors	153
168	Table 34 – wsa:MessageInformationHeaderRequired	154
169	Table 35 – wsman:NoAck	154
170	Table 36 – wsman:QuotaLimit	154
171	Table 37 – wsman:SchemaValidationError	155

172	Table 38 – wsman:TimedOut	155
173	Table 39 – wsman:TimedOut	155
174	Table 40 – wsme:UnableToRenew	156
175	Table 41 – wsme:UnsupportedExpirationType	156
176	Table 42 – wsman:UnsupportedExpirationType	156
177	Table 43 – wsman:UnsupportedFeature	157
178	Table 44 – wsme:UnsupportedExpirationType	158
179	Table 45 – wsman:UnableToRenew	158
180	Table 46 – wsa:InvalidMessage	158
181	Table 47 – wsme:CannotProcessFilter	159
182	Table A-1 – Prefixes and XML Namespaces Used in This Specification	161
183	Table C-1 – Basic Authentication Sequence	165
184	Table C-2 – Digest Authentication Sequence	166
185	Table C-3 – Basic Authentication over HTTPS Sequence	166
186	Table C-4 – Digest Authentication over HTTPS Sequence	167
187	Table C-5 – HTTPS with Client Certificate Sequence	167
188	Table C-6 – Basic Authentication over HTTPS with Client Certificate Sequence	168
189	Table C-7 – SPNEGO Authentication over HTTPS Sequence	169
190	Table C-8 – SPNEGO Authentication over HTTPS with Client Certificate Sequence	169
191	Table D-1 – XPath Level 1 Terminals	173
192	Table D-2 – XPath Level 2 Terminals	175
193		

194

Foreword

195 The *Web Services for Management (WS-Management) Specification* (DSP0226) was prepared by the
 196 WS-Management sub-group of the WBEM Infrastructure & Protocols Working Group.

197 This International Standard makes use of functionality similar to the following W3C
 198 Recommendations:

- 199 • Web Services Eventing (WS-Eventing)
- 200 • Web Services Transfer (WS-Transfer)
- 201 • Web Services Enumeration (WS-Enumeration)

202 These W3C Recommendations were not available at the time WS-Management was defined, and
 203 similar functionality was incorporated directly into provisions of the WS-Management specification.
 204 Future revisions of WS-Management might incorporate these functions by External Reference to
 205 these W3C Recommendations

206 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and
 207 systems management and interoperability.

208 **Acknowledgements**

209 The authors wish to acknowledge the following people.

210 **Chairpersons:**

- 211 • Josh Cohen – Microsoft
- 212 • Larry Lamers (Vice-Chairman) – VMware

213 **Editors:**

- 214 • Nathan Burkhart – Microsoft
- 215 • Doug Davis – IBM
- 216 • Raymond McCollum – Microsoft
- 217 • Bryan Murray – HP
- 218 • Brian Reistad – Microsoft

219 **Authors:**

- 220 • Akhil Arora – Sun Microsystems
- 221 • Vince Brunssen – IBM
- 222 • Mark Carlson – Sun Microsystems
- 223 • Jim Davis – WBEM Solutions
- 224 • Tony Dicenzo – Oracle
- 225 • Mike Dutch – Symantec
- 226 • Zulah Eckert – BEA Systems
- 227 • George Ericson – EMC
- 228 • Wassim Fayed – Microsoft
- 229 • Chris Ferris – IBM
- 230 • Bob Freund – Hitachi Ltd.
- 231 • Eugene Golovinsky – BMC Software
- 232 • Yasuhiro Hagiwara – NEC

- 233 • Steve Hand – Olocity
- 234 • Jackson He – Intel
- 235 • David Hines – Intel
- 236 • Reiji Inohara – NEC
- 237 • Christane Kämpfe – Fujitsu-Siemens Computers
- 238 • Paul Knight – Nortel Networks
- 239 • Vincent Kowalski – BMC Software
- 240 • Heather Kreger – IBM
- 241 • Vishwa Kumbalimutt – Microsoft
- 242 • Sunil Kunisetty – Oracle
- 243 • Richard Landau – Dell
- 244 • Paul Lipton – CA
- 245 • James Martin – Intel
- 246 • Milan Milenkovic – Intel
- 247 • Jeff Mischkinsky – Oracle
- 248 • Paul Montgomery – AMD
- 249 • Jishnu Mukurji – HP
- 250 • Alexander Nosov – Microsoft
- 251 • Abhay Padlia – Novell
- 252 • Gilbert Pilz – Oracle
- 253 • Roger Reich – Symantec
- 254 • Larry Russon – Novell
- 255 • Tom Rutt – Fujitsu Ltd.
- 256 • Jeffrey Schlimmer – Microsoft
- 257 • Dr. Hemal Shah – Broadcom
- 258 • Sharon Smith – Intel
- 259 • Enoch Suen – Dell
- 260 • Vijay Tewari – Intel
- 261 • William Vambenepe – HP
- 262 • Andrea Westerinen – CA, Inc.
- 263 • Kirk Wilson – CA, Inc.
- 264 • Dr. Jerry Xie – Intel

265 Contributors:

- 266 • Paul C. Allen – Microsoft
- 267 • Rodrigo Bomfim – Microsoft
- 268 • Don Box – Microsoft
- 269 • Jerry Duke – Intel
- 270 • David Filani – Intel
- 271 • Kirill Gavrylyuk – Microsoft
- 272 • Omri Gazitt – Microsoft
- 273 • Frank Gorishek – AMD
- 274 • Lawson Guthrie – Intel
- 275 • Arvind Kumar – Intel
- 276 • Brad Lovering – Microsoft

- 277 • Pat Maynard – Intel
- 278 • Steve Millet – Microsoft
- 279 • Matthew Senft – Microsoft
- 280 • Barry Shilmover – Microsoft
- 281 • Tom Slaight – Intel
- 282 • Marvin Theimer – Microsoft
- 283 • Dave Tobias – AMD
- 284 • John Tollefsrud – Sun
- 285 • Anders Vinberg – Microsoft
- 286 • Megan Wallent – Microsoft

287
288

Web Services for Management (WS-Management) Specification

1 Scope

The *Web Services for Management (WS-Management) Specification* describes a Web services protocol based on SOAP for use in management-specific domains. These domains include the management of entities such as PCs, servers, devices, Web services and other applications, and other manageable entities. Services can expose only a WS-Management interface or compose the WS-Management service interface with some of the many other Web service specifications.

A crucial application for these services is in the area of systems management. To promote interoperability between management applications and managed resources, this specification identifies a core set of Web service specifications and usage requirements that expose a common set of operations central to all systems management. This includes the ability to do the following:

- Get, put (update), create, and delete individual resource instances, such as settings and dynamic values
- Enumerate the contents of containers and collections, such as large tables and logs
- Subscribe to events emitted by managed resources
- Execute specific management methods with strongly typed input and output parameters

In each of these areas of scope, this specification defines minimal implementation requirements for conformant Web service implementations. An implementation is free to extend beyond this set of operations, and to choose not to support one or more of the preceding areas of functionality if that functionality is not appropriate to the target device or system.

This specification intends to meet the following requirements:

- Constrain Web services protocols and formats so that Web services can be implemented with a small footprint in both hardware and software management services.
- Define minimum requirements for compliance without constraining richer implementations.
- Ensure backward compatibility and interoperability with WS-Management version 1.0.
- Ensure composability with other Web services specifications.

2 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IETF RFC 2616, R. Fielding et al, *Hypertext Transfer Protocol (HTTP 1.1)*, June 1999, <http://www.ietf.org/rfc/rfc2616.txt>

IETF RFC 2818, E. Rescorla, *HTTP over TLS (HTTPS)*, May 2000, <http://www.ietf.org/rfc/rfc2818.txt>

IETF, RFC 3986, T. Berners-Lee et al, *Uniform Resource Identifiers (URI): Generic Syntax*, August 1998, <http://www.ietf.org/rfc/rfc3986.txt>

- 323 IETF, RFC 4122, P. Leach et al, *A Universally Unique Identifier (UUID) URN Namespace*, July 2005,
 324 <http://www.ietf.org/rfc/rfc4122.txt>
- 325 IETF RFC 4178, L. Zhu et al, *The Simple and Protected Generic Security Service Application*
 326 *Program Interface (GSS-API) Negotiation Mechanism*, October 2005,
 327 <http://www.ietf.org/rfc/rfc4178.txt>
- 328 IETF, RFC 4559, K. Jaganathan et al, *SPNEGO-based Kerberos and NTLM HTTP Authentication in*
 329 *Microsoft Windows*, June 2006, <http://www.ietf.org/rfc/rfc4559.txt>
- 330 IETF RFC 5646, A. Phillips et al, *Tags for Identifying Languages*, September 2009,
 331 <http://tools.ietf.org/rfc/rfc5646.txt>
- 332 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*,
 333 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>
- 334 OASIS, A. Nadalin et al, *Web Services Security Username Token Profile 1.0*, March 2004,
 335 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>
- 336 OASIS, A. Nadalin et al, *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)*,
 337 March 2004, [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
 338 [1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
- 339 OASIS, S. Anderson et al, *Web Services Trust Language (WS-Trust)*, December 2005,
 340 <http://schemas.xmlsoap.org/ws/2005/02/trust>
- 341 The Unicode Consortium, *The Unicode Standard Version 3.0*, January 2000,
 342 <http://www.unicode.org/book/u2.html>
- 343 The Unicode Consortium, *Byte Order Mark (BOM) FAQ*,
 344 http://www.unicode.org/faq/utf_bom.html#BOM
- 345 W3C, M. Gudgin, et al, *SOAP Version 1.2 Part 1: Messaging Framework*, June 2003,
 346 <http://www.w3.org/TR/soap12-part1/>
- 347 W3C, M. Gudgin, et al, *SOAP Version 1.2 Part 2: Adjuncts*, June 2003,
 348 <http://www.w3.org/TR/2003/REC-soap12-part2-20030624>
- 349 W3C, M. Gudgin, et al, *SOAP Message Transmission Optimization Mechanism (MTOM)*,
 350 November 2004, <http://www.w3.org/TR/2004/PR-soap12-mtom-20041116/>
- 351 W3C, J. Clark et al, *XML Path Language Version 1.0 (XPath 1.0)*, November 1999,
 352 <http://www.w3.org/TR/1999/REC-xpath-19991116>
- 353 W3C, J. Cowan et al, *XML Information Set Second Edition (XML Infoset)*, February 2004,
 354 <http://www.w3.org/TR/2004/REC-xml-infoset-20040204/>
- 355 W3C, H. Thompson et al, *XML Schema Part 1: Structures (XML Schema 1)*, May 2001,
 356 <http://www.w3.org/TR/xmlschema-1/>
- 357 W3C, P. Biron et al, *XML Schema Part 2: Datatypes (XML Schema 2)*, May 2001,
 358 <http://www.w3.org/TR/xmlschema-2/>
- 359 W3C, *Web Services Addressing 1.0 – Core*, W3C Recommendation, May 2006,
 360 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
- 361 W3C, *Web Services Addressing 1.0 – SOAP Binding*, W3C Recommendation, May 2006,
 362 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>
- 363 W3C, *Web Services Addressing 1.0 – Metadata*, W3C Recommendation, September 2007,
 364 <http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/>
- 365 W3C, *Extensible Markup Language (XML) 1.0*, W3C Recommendation, October 2000,
 366 <http://www.w3.org/TR/2000/REC-xml-20001006>

- 367 W3C, *Namespaces in XML*, W3C Recommendation, January 1999,
368 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
- 369 W3C, E. Christensen et al, *Web Services Description Language Version 1.1 (WSDL/1.1)*, March
370 2001, <http://www.w3.org/TR/wsdl>
- 371 W3C, S. Boag et al, *XQuery 1.0: An XML Query Language (XQuery 1.0)*, January 2007,
372 <http://www.w3.org/TR/2007/REC-xquery-20070123/>

373 **3 Terms and Definitions**

374 For the purposes of this document, the following terms and definitions apply. The fact that a
375 normative term such as "shall", "shall not", "should", "should not", "may", or "need not" may be used in
376 text which does not have an associated rule number does not mean that the text is not normative.

377 **3.1**

378 **can**

379 used for statements of possibility and capability, whether material, physical, or causal

380 **3.2**

381 **cannot**

382 used for statements of possibility and capability, whether material, physical, or causal

383 **3.3**

384 **conditional**

385 indicates requirements to be followed strictly to conform to the document when the specified
386 conditions are met

387 **3.4**

388 **mandatory**

389 indicates requirements to be followed strictly to conform to the document and from which no deviation
390 is permitted

391 **3.5**

392 **may**

393 indicates a course of action permissible within the limits of the document

394 **3.6**

395 **need not**

396 indicates a course of action permissible within the limits of the document

397 **3.7**

398 **optional**

399 indicates a course of action permissible within the limits of the document

400 **3.8**

401 **shall**

402 indicates requirements to be followed strictly to conform to the document and from which no deviation
403 is permitted

404 **3.9**

405 **shall not**

406 indicates requirements to be followed strictly to conform to the document and from which no deviation
407 is permitted

- 408 **3.10**
 409 **should**
 410 indicates that among several possibilities, one is recommended as particularly suitable, without
 411 mentioning or excluding others, or that a certain course of action is preferred but not necessarily
 412 required
- 413 **3.11**
 414 **should not**
 415 indicates that a certain possibility or course of action is deprecated but not prohibited
- 416 **3.12**
 417 **client**
 418 the application that uses the Web services defined in this document to access the management
 419 service
- 420 **3.13**
 421 **consumer**
 422 the Web service that is requesting the data enumeration from the data source
- 423 **3.14**
 424 **data source**
 425 a Web service that supports traversal using enumeration contexts via the Enumerate operation
 426 defined in this specification
- 427 **3.15**
 428 **delivery mode**
 429 the mechanism by which notification messages are delivered from the source to the sink
- 430 **3.16**
 431 **enumeration context**
 432 a session context that represents a specific traversal through a logical sequence of XML element
 433 information items using the Pull operation defined in this specification
- 434 **3.17**
 435 **event sink**
 436 a Web service that receives notifications
- 437 **3.18**
 438 **event source**
 439 a Web service that sends notifications and accepts requests to create subscriptions
- 440 **3.19**
 441 **managed resource**
 442 an entity that can be of interest to an administrator
 443 It may be a physical object, such as a laptop computer or a printer, or an abstract entity, such as a
 444 service.
- 445 **3.20**
 446 **notification**
 447 a message sent to indicate that an event has occurred

448 **3.21**449 **push mode**

450 a delivery mechanism where the source sends event messages to the sink as individual, unsolicited
451 SOAP messages

452 **3.22**453 **resource**

454 a Web service that is addressable by an endpoint reference and accessed using the operations
455 defined in this specification. This resource can be represented by an XML document. The XML
456 document may be a representation of managed resource

457 **3.23**458 **resource class**

459 an abstract representation (type) of a managed resource

460 A resource class defines the representation of management-related operations and properties. An
461 example of a resource class is the description of operations and properties for a set of laptop
462 computers.

463 **3.24**464 **resource factory**

465 a Web service that is capable of creating new resources using the Create operation defined in this
466 specification

467 **3.25**468 **resource instance**

469 an instantiation of a resource class

470 An example is the set of management-related operations and property values for a specific laptop
471 computer.

472 **3.26**473 **selector**

474 a resource-relative name and value pair that acts as an instance-level discriminant when used with
475 the WS-Management default addressing model

476 A selector is essentially a filter or "key" that identifies the desired instance of the resource. A selector
477 may not be present when service-specific addressing models are used.

478 The relationship of services to resource classes and instances is as follows:

- 479 • A service consists of one or more resource classes.
- 480 • A resource class may contain zero or more instances.

481 If more than one instance for a resource class exists, they are isolated or identified through parts of
482 the SOAP address for the resource, such as the ResourceURI and SelectorSet fields in the default
483 addressing model.

484 **3.27**485 **service**

486 an application that provides management services to clients by exposing the Web services defined in
487 this document

488 Typically, a service is equivalent to the network "listener," is associated with a physical transport
489 address, and is essentially a type of manageability access point.

490 **3.28**491 **subscriber**

492 a Web service that sends requests to create, renew, and/or delete subscriptions

493 **3.29**
 494 **subscription manager**
 495 a Web service that accepts requests to manage, get the status of, renew, and/or delete subscriptions
 496 on behalf of an event source

497 **4 Symbols and Abbreviated Terms**

498 The following symbols and abbreviations are used in this document.

499 **4.1**
 500 **BNF**
 501 Backus-Naur Form (<http://foldoc.org/foldoc/?Backus-Naur+Form>)

502 **4.2**
 503 **BOM**
 504 byte-order mark

505 **4.3**
 506 **CQL**
 507 CIM Query Language

508 **4.4**
 509 **EPR**
 510 Endpoint Reference

511 **4.5**
 512 **GSSAPI**
 513 Generic Security Services Application Program Interface

514 **4.6**
 515 **SOAP**
 516 Simple Object Access Protocol

517 **4.7**
 518 **SPNEGO**
 519 Simple and Protected GSSAPI Negotiation Mechanism

520 **4.8**
 521 **SQL**
 522 Structured Query Language

523 **4.9**
 524 **URI**
 525 Uniform Resource Identifier

526 **4.10**
 527 **URL**
 528 Uniform Resource Locator

529 **4.11**
 530 **UTF**
 531 UCS Transformation Format

532 **4.12**533 **UUID**

534 Universally Unique Identifier

535 **4.13**536 **WSDL**

537 Web Services Description Language

538 **4.14**539 **WS-Man**

540 Web Services Management

541

5 Addressing

542 WS-Management relies on a SOAP-based addressing mechanism (like the one defined in 5.1) to
543 define references to other Web service endpoints and to define some of the headers used in SOAP
544 messages. This addressing mechanism is semantically equivalent and fully wire-compatible with the
545 version of WS-Addressing referenced in WS-Management 1.0. Therefore, this change to WS-
546 Management is fully backward compatible with existing WS-Management implementations.

547 Clause 5.2 specifies how more than one addressing version may be used with WS-Management,
548 such as the version defined in 5.1 or the W3C Recommendation version of addressing. In this
549 specification, unless explicitly referring to a particular version, the term "Addressing" refers generically
550 to either version of addressing as defined in 5.2.

551 Multiple addressing models may be used with any of the addressing versions described in 5.2.
552 Implementations may implement any of the following addressing models:

- 553 • basic addressing as defined in 5.1
- 554 • the Default Addressing Model as defined in 5.4.2
- 555 • new addressing models that are not defined in this specification. These addressing models
556 may impose additional restrictions or requirements for addressing.

557

5.1 Management Addressing

558 The features defined in this clause provide a transport-neutral mechanism to address Web services
559 and messages. Specifically, this clause defines XML elements to identify Web service endpoints and
560 to secure end-to-end endpoint identification in messages. This enables messaging systems to
561 support message transmission through networks that include processing nodes such as endpoint
562 managers, firewalls, and gateways in a transport-neutral manner.

563

5.1.1 Introduction

564 This clause defines two interoperable constructs, endpoint references and message information
565 headers, that convey information that is typically provided by transport protocols and messaging
566 systems. These constructs normalize this underlying information into a uniform format that can be
567 processed independently of transport or application.

568 A Web service endpoint is an entity, processor, or resource that can be referenced and can be
569 targeted for Web service messages. Endpoint references convey the information needed to identify
570 and reference a Web service endpoint, and they may be used in several different ways:

- 571 • Endpoint references are suitable for conveying the information needed to access a Web
572 service endpoint.

- 573 • Endpoint references are also used to provide addresses for individual messages sent to
574 and from Web services.

575 To deal with the latter use case, this clause defines a family of message information headers that
576 allows uniform addressing of messages independent of underlying transport. These message
577 information headers convey end-to-end message characteristics including addressing for source and
578 destination endpoints as well as message identity.

579 EXAMPLE: The following example illustrates the use of these mechanisms in a SOAP 1.2 message being sent
580 from <http://business456.example/client1> to <http://fabrikam123.example/Purchasing>.

581 Lines (002) to (014) represent the header of the SOAP message where the mechanisms defined in this clause
582 are used. The body is represented by lines (015) to (017).

583 Lines (003) to (013) contain the message information header blocks. Specifically, lines (003) to (005) specify the
584 identifier for this message, lines (006) to (008) specify the endpoint from where the message originated, and
585 lines (009) to (011) specify the endpoint to which replies to this message should be sent as an Endpoint
586 Reference. Line (012) specifies the address URI of the ultimate receiver of this message. Line (013) specifies an
587 Action URI identifying expected semantics.

```
588 (001) <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
589         xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
590 (002)   <S:Header>
591 (003)     <wsa:MessageID>
592 (004)       uuid:6B29FC40-CA47-1067-B31D-00DD010662DA
593 (005)     </wsa:MessageID>
594 (006)     <wsa:From>
595 (007)       <wsa:Address>http://business456.example/client1</wsa:Address>
596 (008)     </wsa:From>
597 (009)     <wsa:ReplyTo>
598 (010)       <wsa:Address>http://business456.example/client1</wsa:Address>
599 (011)     </wsa:ReplyTo>
600 (012)     <wsa:To>http://fabrikam123.example/Purchasing</wsa:To>
601 (013)     <wsa:Action>http://fabrikam123.example/SubmitPO</wsa:Action>
602 (014)   </S:Header>
603 (015)   <S:Body>
604 (016)     ...
605 (017)   </S:Body>
606 (018) </S:Envelope>
```

607 5.1.2 Endpoint References

608 This clause defines the syntax of an Endpoint Reference (EPR).

609 5.1.2.1 Format of Endpoint References

610 This clause defines an XML representation for an endpoint reference as both an XML type
611 (`wsa:EndpointReferenceType`) and as an XML element (`<wsa:EndpointReference>`).

612 The `wsa:EndpointReferenceType` type is used wherever a Web service endpoint is referenced. The
613 following describes the contents of this type:

```
614 <wsa:EndpointReference>
615   <wsa:Address>xs:anyURI</wsa:Address>
616   <wsa:ReferenceProperties>... </wsa:ReferenceProperties> ?
617   <wsa:ReferenceParameters>... </wsa:ReferenceParameters> ?
618   <wsa:PortType>xs:QName</wsa:PortType> ?
619   <wsa:ServiceName PortName="xs:NCName"?>xs:QName</wsa:ServiceName> ?
620   <wsp:Policy> ... </wsp:Policy>*
621 </wsa:EndpointReference>
```

622 The following describes the attributes and elements listed in the preceding schema overview:

623 `wsa:EndpointReference`

624 This represents some element of type `wsa:EndpointReferenceType`. This example uses the
625 predefined `<wsa:EndpointReference>` element, but any element of type
626 `wsa:EndpointReferenceType` may be used.

627 `wsa:EndpointReference/wsa:Address`

628 This required element (of type `xs:anyURI`) specifies the address URI that identifies the endpoint.
629 This address may be a logical address or identifier for the service endpoint.

630 `wsa:EndpointReference/wsa:ReferenceProperties/`

631 This optional element contains any number of individual reference properties that are associated
632 with the endpoint to facilitate a particular interaction. Reference properties are XML elements that
633 are required to properly interact with the endpoint. Reference properties are provided by the issuer
634 of the endpoint reference and are otherwise assumed to be opaque to consuming applications.

635 NOTE: The use of reference properties is deprecated; reference parameters should be used instead.

636 `wsa:EndpointReference/wsa:ReferenceProperties/{any}`

637 Each child element of `ReferenceProperties` represents an individual reference property.

638 `wsa:EndpointReference/wsa:ReferenceParameters/`

639 This optional element contains any number of individual parameters that are associated with the
640 endpoint to facilitate a particular interaction. Reference parameters are XML elements that are
641 required to properly interact with the endpoint. Reference parameters are also provided by the
642 issuer of the endpoint reference and are otherwise assumed to be opaque to consuming
643 applications.

644 See 5.4 for some WS-Management-specific reference parameters.

645 `wsa:EndpointReference/wsa:ReferenceParameters/{any}`

646 Each child element of `ReferenceParameters` represents an individual reference parameter.

647 `wsa:EndpointReference/wsa:PortType`

648 This optional element (of type `xs:QName`) specifies the value of the primary portType of the
649 endpoint being conveyed.

650 NOTE: The use of `wsa:PortType` is deprecated.

651 `wsa:EndpointReference/wsa:ServiceName`

652 This optional element (of type `xs:QName`) specifies the `<wsdl:service>` definition that contains a
653 WSDL description of the endpoint being referenced. The service name provides a link to a full
654 description of the service endpoint. An optional non-qualified name identifies the specific port in
655 the service that corresponds to the endpoint.

656 NOTE: The use of `wsa:ServiceName` is deprecated.

657 `wsa:EndpointReference/wsa:ServiceName/@PortName`

658 This optional attribute (of type `xs:NCName`) specifies the name of the `<wsdl:port>` definition that
659 corresponds to the endpoint being referenced.

660 `wsa:EndpointReference/wsp:Policy`

661 This optional element specifies a policy that is relevant to the interaction with the endpoint.

662 NOTE: The use of `wsp:Policy` is deprecated.

663 `wsa:EndpointReference/{any}`

664 This is an extensibility mechanism to allow additional elements to be specified.

665 `wsa:EndpointReference/@{any}`

666 This is an extensibility mechanism to allow additional attributes to be specified.

667 EXAMPLE: The following example illustrates an endpoint reference. This element references the URI
668 "http://www.fabrikam123.example/acct":

```
669 <wsa:EndpointReference xmlns:wsa="..." xmlns:fabrikam="...">
670   <wsa:Address>http://www.fabrikam123.example/acct</wsa:Address>
671 </wsa:EndpointReference>
```

672 5.1.2.2 Binding Endpoint References

673 When a message needs to be addressed to the endpoint, the information contained in the endpoint
674 reference is mapped to the message according to a transformation that is dependent on the protocol
675 and data representation used to send the message. Protocol-specific mappings (or bindings) define
676 how the information in the endpoint reference is copied to message and protocol fields. This clause
677 defines the SOAP binding for endpoint references. This mapping may be explicitly replaced by other
678 bindings (defined as WSDL bindings or as policies); however, in the absence of an applicable policy
679 stating that a different mapping is to be used, the SOAP binding defined here is assumed to apply. To
680 ensure interoperability with a broad range of devices, all conformant implementations shall support
681 the SOAP binding.

682 The SOAP binding for endpoint references is defined by the following two rules:

683 **R5.1.2.2-1:** The `wsa:Address` element in the endpoint reference shall be copied in the `wsa:To`
684 header field of the SOAP message.

685 **R5.1.2.2-2:** Each Reference Property and Reference Parameter element becomes a header
686 block in the SOAP message. The elements of each Reference Property or Reference Parameter
687 (including all of its child elements, attributes, and in-scope namespaces) shall be added as a
688 header block in the new message.

689 EXAMPLE: The following example shows how the default SOAP binding for endpoint references is used to
690 construct a message addressed to the endpoint:

```
691 <wsa:EndpointReference xmlns:wsa="..." xmlns:fabrikam="...">
692   <wsa:Address>http://www.fabrikam123.example/acct</wsa:Address>
693   <wsa:ReferenceParameters>
694     <fabrikam:CustomerKey>123456789</fabrikam:CustomerKey>
695     <fabrikam:ShoppingCart>ABCDEFGF</fabrikam:ShoppingCart>
696   </wsa:ReferenceParameters>
697 </wsa:EndpointReference>
```

698 According to the mapping rules stated before, the address value is copied in the "To" header and the
699 "CustomerKey" element should be copied literally as a header in a SOAP message addressed to this
700 endpoint. The SOAP message would look as follows:

```
701 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
702   xmlns:wsa="..." xmlns:fabrikam="...">
703   <S:Header>
```

```

704     ...
705     <wsa:To>http://www.fabrikam123.example/acct</wsa:To>
706     <fabrikam:CustomerKey>123456789</fabrikam:CustomerKey>
707     <fabrikam:ShoppingCart>ABCDEFG</fabrikam:ShoppingCart>
708     ...
709     </S:Header>
710     <S:Body>
711     ...
712     </S:Body>
713 </S:Envelope>

```

714 5.1.3 Message Information Headers

715 This clause defines the syntax of a message information header.

716 The message information headers collectively augment a message with the headers shown in
 717 Figure 1. These headers enable the identification and location of the endpoints involved in an
 718 interaction. The basic interaction pattern from which all others are composed is "one way". In this
 719 pattern a source sends a message to a destination without any further definition of the interaction.

720 "Request Reply" is a common interaction pattern that consists of an initial message sent by a source
 721 endpoint (the request) and a subsequent message sent from the destination of the request back to
 722 the source (the reply). A reply can be an application message, a fault, or any other message.

723 The message information header blocks provide end-to-end characteristics of a message that can be
 724 easily secured as a unit. The information in these headers is immutable and not intended to be
 725 modified along the message path.

726 Figure 1 shows the contents of the message information header blocks:

```

727 <wsa:MessageID> xs:anyURI </wsa:MessageID>
728 <wsa:RelatesTo RelationshipType="..."?>xs:anyURI</wsa:RelatesTo>
729 <wsa:To>xs:anyURI</wsa:To>
730 <wsa:Action>xs:anyURI</wsa:Action>
731 <wsa:From>endpoint-reference</wsa:From>
732 <wsa:ReplyTo>endpoint-reference</wsa:ReplyTo>
733 <wsa:FaultTo>endpoint-reference</wsa:FaultTo>

```

734 **Figure 1 – Message Information Header Blocks**

735 The following describes the attributes and elements listed in Figure 1:

736 **wsa:MessageID**

737 This optional element (of type xs:anyURI) uniquely identifies this message in time and space. This
 738 element shall be present if wsa:ReplyTo or wsa:FaultTo is present. No two messages with a
 739 distinct application intent may share a wsa:MessageID value. A message may be retransmitted for
 740 any purpose (including communications failure) and may use the same wsa:MessageID value.
 741 The value of this header is an opaque URI whose interpretation beyond equivalence is not defined
 742 in this specification. If a reply is expected, this property shall be present.

743 wsa:RelatesTo

744 This optional (repeating) element indicates how this message relates to another message, in the
 745 form of a URI-QName pair. The child of this element (which is of type xs:anyURI) contains the
 746 wsa:MessageID of the related message or the following well-known URI that means "unspecified
 747 message":

748 `http://schemas.xmlsoap.org/ws/2004/08/addressing/id/unspecified`

749 A reply message shall contain a wsa:RelatesTo header consisting of wsa:Reply and the
 750 wsa:MessageID value of the request message.

751 wsa:RelatesTo/@RelationshipType

752 This optional attribute (of type xs:QName) conveys the relationship type as a QName. When
 753 absent, the implied value of this attribute is wsa:Reply.

754 This specification has one predefined relationship type, as shown in Table 1:

755 **Table 1 – Relationship Type**

QName	Description
wsa:Reply	Indicates that this is a reply to the message identified by the URI.

756 wsa:ReplyTo

757 This optional element (of type wsa:EndpointReferenceType) provides an endpoint reference that
 758 identifies the intended receiver for replies to this message. This element shall be present if a reply
 759 is expected. If this element is present, wsa:MessageID shall be present. If a reply is expected, a
 760 message shall contain a wsa:ReplyTo header. The sender shall use the contents of the
 761 wsa:ReplyTo to formulate the reply message as defined in 5.1.3.1. If the wsa:ReplyTo header is
 762 absent, the contents of the wsa:From header may be used to formulate a message to the source.
 763 This header may be absent if the message has no meaningful reply.

764 wsa:From

765 This optional element (of type wsa:EndpointReferenceType) provides a reference to the endpoint
 766 where the message originated.

767 wsa:FaultTo

768 This optional element (of type wsa:EndpointReferenceType) provides an endpoint reference that
 769 identifies the intended receiver for faults related to this message. If this element is present,
 770 wsa:MessageID shall be present. When formulating a fault message as defined in 5.1.3.1, the
 771 sender shall use the contents of this header to formulate the fault message. If this header is
 772 absent, the sender should use the contents of the wsa:ReplyTo header to formulate the fault
 773 message. If both the wsa:FaultTo and wsa:ReplyTo header are absent, the sender may use the
 774 contents of the wsa:From header to formulate the fault message.

775 wsa:To

776 This required element (of type xs:anyURI) provides the address of the intended receiver of this
 777 message.

778 wsa:Action

779 This required element (of type xs:anyURI) uniquely identifies the semantics implied by this
 780 message. It is recommended that the value of this header be a URI identifying an input, output, or
 781 fault message within a WSDL port type. An action may be explicitly or implicitly associated with

782 the corresponding WSDL definition. Finally, if in addition to the wsa:Action header, a SOAP Action
 783 URI is encoded in a request, the URI of the SOAP Action shall either be the same as the one
 784 specified by the wsa:Action header, or set to "".

785 The dispatching of incoming messages is based on two message properties. The mandatory wsa:To
 786 and wsa:Action header identify the target processing location and the verb or intent of the message.

787 Due to the range of network technologies currently in wide-spread use (for example, NAT, DHCP,
 788 and firewalls), many deployments cannot assign a meaningful global URI to a given endpoint. To
 789 allow these "anonymous" endpoints to initiate message exchange patterns and receive replies,
 790 Addressing defines the following well-known URI for use by endpoints that cannot have a stable,
 791 resolvable URI:

792 `http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous`

793 Requests whose wsa:ReplyTo, wsa:From and/or wsa:FaultTo headers use this address shall provide
 794 some out-of-band mechanism for delivering replies or faults (for example, returning the reply on the
 795 same transport connection). This mechanism may be a simple request/reply transport protocol (for
 796 example, HTTP GET or POST). This URI may be used as the wsa:To header for reply messages and
 797 should not be used as the wsa:To header in other circumstances.

798 5.1.3.1 Formulating a Reply Message

799 The reply to an Addressing compliant request message shall be constructed according to the rules
 800 defined in this clause.

801 EXAMPLE 1: The following example illustrates a request message using message information header blocks in a
 802 SOAP 1.2 message:

```
803 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
804   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
805   xmlns:f123="http://www.fabrikam123.example/svc53">
806   <S:Header>
807     <wsa:MessageID>uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff
808     </wsa:MessageID>
809     <wsa:ReplyTo>
810       <wsa:Address>http://business456.example/client1</wsa:Address>
811     </wsa:ReplyTo>
812     <wsa:To S:mustUnderstand="1">mailto:joe@fabrikam123.example</wsa:To>
813     <wsa:Action>http://fabrikam123.example/mail/Delete</wsa:Action>
814   </S:Header>
815   <S:Body>
816     <f123:Delete>
817       <maxCount>42</maxCount>
818     </f123:Delete>
819   </S:Body>
820 </S:Envelope>
```

821 EXAMPLE 2: The following example illustrates a reply message using message information header blocks in a
 822 SOAP 1.2 message:

```
823 <S:Envelope
824   xmlns:S="http://www.w3.org/2003/05/soap-envelope"
825   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
826   xmlns:f123="http://www.fabrikam123.example/svc53">
827   <S:Header>
828     <wsa:MessageID>
829       uuid:aaaabbbb-cccc-dddd-eeee-wwwwwwwwwwww
830     </wsa:MessageID>
831     <wsa:RelatesTo>
```

```

832     uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff
833   </wsa:RelatesTo>
834   <wsa:To>
835     http://business456.example/client1
836   </wsa:To>
837   <wsa:Action>http://fabrikam123.example/mail/DeleteAck</wsa:Action>
838 </S:Header>
839 <S:Body>
840   <f123>DeleteAck/>
841 </S:Body>
842 </S:Envelope>

```

5.1.3.2 Associating Action with WSDL Operations

Addressing defines two mechanisms, explicit association and default action pattern, to associate an action with input, output, and fault elements within a WSDL port type.

5.1.3.2.1 Explicit Association

The action may be explicitly associated using the `wsa:Action` attribute.

EXAMPLE: Consider the following WSDL excerpt:

```

849 <definitions targetNamespace="http://example.com/stockquote" ...>
850   ...
851   <portType name="StockQuotePortType">
852     <operation name="GetLastTradePrice">
853       <input message="tns:GetTradePricesInput"
854         wsa:Action="http://example.com/GetQuote"/>
855       <output message="tns:GetTradePricesOutput"
856         wsa:Action="http://example.com/Quote"/>
857     </operation>
858   </portType>
859   ...
860 </definitions>

```

The action for the input of the `GetLastTradePrice` operation within the `StockQuotePortType` is explicitly defined to be `http://example.com/GetQuote`. The action for the output of this same operation is `http://example.com/Quote`.

5.1.3.2.2 Default Action Pattern

In the absence of the `wsa:Action` attribute, the following pattern is used to construct a default action for inputs and outputs. The general form of an action URI is as follows:

```
targetNamespace/portTypeName/(inputName|outputName)
```

The `"/"` is a literal character to be included in the action. The values of the properties are as follows:

- *targetNamespace* is the target namespace (`/definition/@targetNamespace`). If *target namespace* ends with a `"/"` an additional `"/"` is not added.
- *portTypeName* is the name of the port type (`/definition/portType/@name`).
- *(inputName|outputName)* is the name of the element as defined in Section 2.4.5 of [WSDL 1.1](#).

For fault messages, this pattern is not applied. Instead, the following URI is the default action URI for fault messages:

```
http://schemas.xmlsoap.org/ws/2004/08/addressing/fault
```

876 EXAMPLE: Consider the following WSDL excerpt:

```
877 <definitions targetNamespace="http://example.com/stockquote" ...>
878   ...
879   <portType name="StockQuotePortType">
880     <operation name="GetLastTradePrice">
881       <input message="tns:GetTradePricesInput" name="GetQuote"/>
882       <output message="tns:GetTradePricesOutput" name="Quote"/>
883     </operation>
884   </portType>
885   ...
886 </definitions>
```

887 *targetNamespace* = http://example.com/stockquote

888 *portTypeName* = StockQuotePortType

889 *inputName* = GetQuote

890 *outputName* = Quote

891 Applying the preceding pattern with these values produces the following:

892 input action = http://example.com/stockquote/StockQuotePortType/GetQuote

893 output action = http://example.com/stockquote/StockQuotePortType/Quote

894 WSDL defines rules for a default input or output name if the name attribute is not present. Consider
895 the following example:

896 EXAMPLE: The following is a WSDL excerpt:

```
897 <definitions targetNamespace="http://example.com/stockquote" ...>
898   ...
899   <portType name="StockQuotePortType">
900     <operation name="GetLastTradePrice">
901       <input message="tns:GetTradePricesInput"/>
902       <output message="tns:GetTradePricesOutput"/>
903     </operation>
904   </portType>
905   ...
906 </definitions>
```

907 *targetNamespace* = http://example.com/stockquote

908 *portTypeName* = StockQuotePortType

909 According to the rules defined in 2.4.5 of [WSDL](#), if the name attribute is absent for the input of a
910 request response operation, the default value is the name of the operation with "Request" appended.

911 *inputName* = GetLastTradePriceRequest

912 Likewise, the output defaults to the operation name with "Response" appended.

913 *outputName* = GetLastTradePriceResponse

914 Applying the previous pattern with these values produces the following:

915 input action = http://example.com/stockquote/StockQuotePortType/GetLastTradePriceRequest

916 output action = http://example.com/stockquote/StockQuotePortType/GetLastTradePriceResponse

5.2 Versions of Addressing

To maintain compatibility with implementations of previous versions of WS-Management, this protocol accommodates messages formatted by those previous versions. However, WS-Management 1.1 also allows for the optional use of the [WS-Addressing W3C Recommendation](#).

The following abbreviations are used for clarity and brevity.

- "WSMA" refers to the version of Management Addressing as specified in 5.1.
- "WSA-Rec" refers to the WS-Addressing W3C Recommendation.
- "WS-Man 1.0" refers to the *WS-Management Specification* 1.0 and implementations compatible with that specification.
- "WS-Man 1.1" refers to this specification and implementations compatible with this specification.
- "Addressing Anonymous URI" refers to the anonymous URI that is defined by the version of Addressing currently in use. The anonymous URI defined by WSA-Rec is <http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous>. The anonymous URI defined by WSMA is <http://www.w3.org/2005/08/addressing/anonymous>.

NOTE: Some information in this clause is implementation advice to clients on algorithms for efficient communication with unknown services. This informative advice should not be construed to place normative requirements on the behavior of compliant clients or services.

5.2.1 Technical Differences

The [WSMA](#) and [WSA-Rec](#) specifications reference different XML namespaces. An endpoint sending Web service messages shall use, for the Addressing SOAP headers, one namespace or the other; a receiving endpoint may recognize one namespace or both namespaces. Existing implementations of WS-Man 1.0 are limited to recognizing only the WSMA namespace. Interactions between WS-Man 1.0 and WS-Man 1.1 implementations will have to allow for these limitations.

5.3 Requirements for Compatibility

To maximize interoperability of WS-Management implementations, WS-Man 1.0 and WS-Man 1.1 clients and services need to be able to exchange messages. These requirements are summarized in Table 2.

Table 2 – Interoperability Requirements

Interoperability Requirements between WS-Management Versions	WS-Man 1.0 Service	WS-Man 1.1 Service
WS-Man 1.0 client	It works.	WS-Man 1.0 client needs to be able to access WS-Man 1.1 service, but some negotiation might be needed.
WS-Man 1.1 client	WSMan 1.1 client needs to be able to access 1.0 service.	It works, but some negotiations might be needed.

Homogeneous pairings of compliant clients and services (that is, a version 1.0 client with a version 1.0 service, or a version 1.1 client with a version 1.1 service) can exchange messages in accordance with their respective specifications. To ensure reliable communications, heterogeneous pairings need to meet certain requirements and implement certain sequencing strategies.

950 In particular, clients and services that implement WS-Man 1.0 can use only WSMA in any exchanges;
 951 therefore, all exchanges with version 1.0 endpoints use only WSMA. This conclusion is summarized
 952 in Table 3.

953 **Table 3 – WSA Versions in Exchanges**

Interoperable Version of Addressing	WS-Man 1.0 Service	WS-Man 1.1 Service
WS-Man 1.0 client	WSMA	WSMA
WS-Man 1.1 client	WSMA	WSMA or WSA-Rec

954 **5.3.1 Discovery or Negotiation**

955 If it is possible for a client to determine the capabilities of the service with respect to WSA, such
 956 discovery is more efficient than negotiating the WSA version. For instance, if a service supports
 957 Identify, then a client can determine in advance the WS-Man protocol, as well as an Addressing
 958 version or versions supported by the service. For this reason, support of Identify is mandatory in this
 959 specification when [WSA-Rec](#) is used.

960 Identify would be used as follows:

- 961 • The client sends the service an Identify message.
- 962 • If the service does not support Identify, the client can conclude that the service is a WS-
 963 Man 1.0 implementation and only supports WSMA.
- 964 • If the service successfully processes the Identify message, the client examines the versions
 965 of Addressing by looking at the AddressingVersionURI element (as defined in clause 11), if
 966 present, and can choose the appropriate version.
- 967 • If the Identify response message does not contain any Addressing versions, then there is
 968 no way for the client to know which version of Addressing to use and it would need to use
 969 one of the strategies described in 5.3.2.

970 In any case, to avoid unnecessary re-discovery or re-negotiation, a WS-Man 1.1 client should retain
 971 information about the capabilities of service endpoints where practical.

972 **5.3.2 Client Negotiation Strategies**

973 A compliant WS-Man 1.0 client will use only WSMA in message exchanges. A WS-Man 1.1 client,
 974 however, may use either WSMA or WSA-Rec in message exchanges. If a WS-Man 1.1 client does
 975 not know the WSA version capabilities of a service, it may use different strategies when initially
 976 contacting the service. The client may begin a message exchange with either version of WSA, using
 977 WSA-Rec or WSMA in the request message. The message exchange would proceed as follows:

- 978 • Strategy type 1: A client sends the request using WSA-Rec. The WSA-Rec SOAP headers
 979 need to be marked with a mustUnderstand="1" attribute to ensure that a fault will be
 980 generated if the receiver does not support the WSA-Rec version of Addressing. The client
 981 can then retry the operation using WSMA.
- 982 • Strategy type 2: A client sends the request using WSMA. Both WS-Man 1.0 services and
 983 WS-Man 1.1 services respond to the request using WSMA.

984 **5.3.3 Initiating Message Exchanges**

985 Outgoing messages initiated by a WS-Man implementation need to use the same version of
 986 Addressing that was used in the Endpoint Reference to which those messages are being sent. For
 987 example, if a Subscribe request message uses WSA-Rec in the SOAP headers (for example, for the
 988 wsa:To and wsa:ReplyTo), but uses WSMA for the NotifyTo EPR, then the Subscribe response will
 989 be sent using WSA-Rec, but the events will be sent using WSMA.

990 5.3.4 Normative Rules

991 **R5.3.4-1:** If a WS-Man 1.1 service supports WSA-Rec, then it shall also support the Identify
 992 operation.

993 **R5.3.4-2:** A WS-Man 1.1 service shall support WSMA and should support WSA-Rec.

994 **R5.3.4-3:** A WS-Man 1.1 implementation shall send messages to endpoints using the same
 995 version of Addressing used in the Endpoint Reference of the destination endpoint (see 5.2).

996 **R5.3.4-4:** Within a single SOAP message, a WS-Man 1.1 implementation shall use the same
 997 version of Addressing for all Addressing SOAP headers.

998 Because WS-Man 1.1 allows for either version of Addressing to be used, R5.3.4-4 removes the
 999 possibility of mixing the two versions for the WSA SOAP headers, but it does not disallow Endpoint
 1000 References that might appear elsewhere in the message to be of a different version.

1001 In order to provide a migration path from the WSMA to WSA-Rec, the schema of certain messages
 1002 allows for either version's EndpointReferenceType to be used. While the schema itself is written in a
 1003 very generic way (that is, using an xs:any) allowing any arbitrary XML to appear, implementations
 1004 shall restrict the contents of this element to one of the EndpointReference Types.

1005 NOTE: This allows existing WS-Man 1.0 implementations to be compliant, while providing newer
 1006 implementations a migration path. In this spirit, newer implementations are strongly encouraged to support both
 1007 versions of Addressing.

1008 5.4 Use of Addressing in WS-Management

1009 This clause describes the use of Endpoint References regardless of whether an implementation uses
 1010 WS-Management Addressing (see 5.1) or the W3C Recommendation version of WS-Addressing.

1011 Addressing (either addressing type) endpoint references (EPRs) are used to convey information
 1012 needed to address a Web service endpoint. WS-Management defines a default addressing model
 1013 that can optionally be used in EPRs.

1014 5.4.1 Use of Endpoint References

1015 WS-Management uses EPRs as the addressing mechanism for individual resource instances.
 1016 WS-Management also defines a default addressing model for use in addressing resources. In cases
 1017 where this default addressing model is not appropriate, such as in systems with well-established
 1018 addressing models or with EPRs retrieved from a discovery service, services may use those service-
 1019 specific addressing models if they are based on either addressing version supported by WS-
 1020 Management.

1021 **R5.4.1-1:** All messages that are addressed to a resource class or instance that is referenced
 1022 by an EPR must follow the Addressing rules for representing content from the EPR (the address
 1023 and reference parameters) in the SOAP message. This rule also applies to continuation
 1024 messages such as Pull or Release, which continue an operation begun in a previous message.
 1025 Even though such messages contain contextual information that binds them to a previous
 1026 operation, the information from the EPR is still required in the message to help route it to the
 1027 correct handler.

1028 Rule R5.4.1-1 clarifies that messages such as Pull or Renew still require a full EPR. For Pull, for
 1029 example, this EPR would be the same as the original Enumerate, even though EnumerateResponse
 1030 returns a context object that would seem to obviate the need for the EPR. The EPR is still required to
 1031 route the message properly. Similarly, the Renew request uses the SubscriptionManager EPR
 1032 received in the SubscribeResponse.

1033 When a service includes an EPR in a response message, it must be willing to accept subsequent
 1034 request messages targeted to that EPR for the same individual managed resource. Clients are not
 1035 required to process or enhance EPRs given to them by the service before using them to address a
 1036 managed resource.

1037 **R5.4.1-2:** An EPR returned by a service shall be acceptable to that service to refer to the
 1038 same managed resource.

1039 **R5.4.1-3:** All EPRs returned by a service, whether expressed using the WS-Management
 1040 default addressing model (see 5.4.2) or any other addressing model, shall be valid as long as the
 1041 managed resource exists.

1042 5.4.2 WS-Management Default Addressing Model

1043 WS-Management defines a default addressing model for resources. A service is not required to use
 1044 this addressing model, but it is suitable for many new implementations and can increase the chances
 1045 of successful interoperation between clients and services.

1046 This document uses examples of this addressing model that contain its component parts, the
 1047 ResourceURI and SelectorSet SOAP headers. This specification is independent of the actual data
 1048 model and does not define the structure of the ResourceURI or the set of values for selectors for a
 1049 given resource. These may be vendor specific or defined by other specifications.

1050 Description and use of this addressing model in this specification do not indicate that support for this
 1051 addressing model is a requirement for a conformant service.

1052 All of the normative text, examples, and conformance rules in 5.4.2 and 5.4.2.2 presume that the
 1053 service is based on the default addressing model. In cases where this addressing model is not in use,
 1054 these rules do not apply.

1055 The default addressing model uses a representation of an EPR that is a tuple of the following SOAP
 1056 headers:

- 1057 • **wsa:To** (required): the transport address of the service
- 1058 • **wsman:ResourceURI** (required if the default addressing model is used): the URI of the
 1059 resource class representation or instance representation
- 1060 • **wsman:SelectorSet** (optional): a header that identifies or "selects" the resource instance to
 1061 be accessed if more than one instance of a resource class exists

1062 The wsman:ResourceURI value needs to be marked with an s:mustUnderstand attribute set to "true"
 1063 in all messages that use the default addressing model. Otherwise, a service that does not understand
 1064 this addressing model might inadvertently return a resource that was not requested by the client.

1065 The WS-Management default addressing model is defined in the following XML outline for an EPR:

```

1066 (1) <wsa:EndpointReference>
1067 (2)   <wsa:Address>
1068 (3)     Network address
1069 (4)   </wsa:Address>
1070 (5)   <wsa:ReferenceParameters>
1071 (6)     <wsman:ResourceURI> resource URI </wsman:ResourceURI>
1072 (7)     <wsman:SelectorSet>
```

```

1073 (8)      <wsman:Selector Name="selector-name"> *
1074 (9)      Selector-value
1075 (10)     </wsman:Selector>
1076 (11)     </wsman:SelectorSet> ?
1077 (12)     </wsa:ReferenceParameters>
1078 (13)     </wsa:EndpointReference>

```

1079 The following definitions provide additional, normative constraints on the preceding outline:

1080 **wsa:Address**

1081 the URI of the transport address

1082 **wsa:ReferenceParameters/wsman:ResourceURI**

1083 the URI of the resource class or instance to be accessed

1084 Typically, this URI represents the resource class, but it may represent the instance. The
 1085 combination of this URI and the wsa:To URI form the full address of the resource class or
 1086 instance.

1087 **wsa:ReferenceParameters/wsman:SelectorSet:**

1088 the optional set of selectors as described in 5.4.2.2

1089 These values are used to select an instance if the ResourceURI identifies a multi-instanced
 1090 target.

1091 When the default addressing model is used in a SOAP message, Addressing specifies that
 1092 translations take place and the headers are flattened out.

1093 **EXAMPLE:** The following is an example EPR definition:

```

1094 (1) <wsa:EndpointReference>
1095 (2) <wsa:Address> Address </wsa:Address>
1096 (3) <wsa:ReferenceParameters xmlns:wsman="...">
1097 (4) <wsman:ResourceURI>resURI</wsman:ResourceURI>
1098 (5) <wsman:SelectorSet>
1099 (6) <wsman:Selector Name="Selector-name">
1100 (7) Selector-value
1101 (8) </wsman:Selector>
1102 (9) </wsman:SelectorSet>
1103 (10) </wsa:ReferenceParameters>
1104 (11) </wsa:EndpointReference>

```

1105 This address definition is translated as follows when used in a SOAP message. wsa:Address becomes wsa:To,
 1106 and the reference parameters are unwrapped and juxtaposed. The following example shows a sample SOAP
 1107 message using WSMA:

```

1108 (1) <s:Envelope xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
1109 (2) <s:Header>
1110 (3) <wsa:To> Address </wsa:To>
1111 (4) <wsa:Action> Action URI </wsa:Action>
1112 (5) <wsman:ResourceURI s:mustUnderstand="true">resURI</wsman:ResourceURI>
1113 (6) <wsman:SelectorSet>
1114 (7) <wsman:Selector Name="Selector-name">
1115 (8) Selector-value
1116 (9) </wsman:Selector>
1117 (10) </wsman:SelectorSet>
1118 (11) ...
1119 (12) </s:Header>
1120 (13) <s:Body> ... </s:Body>
1121 (14) </s:Envelope>

```

1122 The following message shows a sample SOAP message using WS-Rec:

```

1123 (1) <s:Envelope xmlns:wsa="http://www.w3.org/2005/08/addressing ">
1124 (2)   <s:Header>
1125 (3)     <wsa:To s:mustUnderstand="true"> Address </wsa:To>
1126 (4)     <wsa:Action s:mustUnderstand="true"> Action URI </wsa:Action>
1127 (5)     <wsman:ResourceURI s:mustUnderstand="true"
1128 (6)       wsa:isReferenceParameter="true">resURI</wsman:ResourceURI>
1129 (7)     <wsman:SelectorSet wsa:isReferenceParameter="true">
1130 (8)       <wsman:Selector Name="Selector-name">
1131 (9)         Selector-value
1132 (10)       </wsman:Selector>
1133 (11)     </wsman:SelectorSet>
1134 (12)   ...
1135 (13) </s:Header>
1136 (14) <s:Body> ... </s:Body>
1137 (15) </s:Envelope>

```

1138 In both cases, the wsa:To, wsman:ResourceURI, and wsman:SelectorSet elements work together to
 1139 *reference* the resource instance to be managed, but the actual *method* or *operation* to be executed
 1140 against this resource is indicated by the wsa:Action header.

1141 **EXAMPLE:** The following is an example of Addressing headers based on the default addressing model in an
 1142 actual message:

```

1143 (1) <s:Envelope
1144 (2)   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
1145 (3)   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
1146 (4)   xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
1147 (5)   <s:Header>
1148 (6)     ...
1149 (7)     <wsa:To>http://123.99.222.36/wsman</wsa:To>
1150 (8)     <wsman:ResourceURI s:mustUnderstand="true">
1151 (9)       http://example.org/hardware/2005/02/storage/physDisk
1152 (10)    </wsman:ResourceURI>
1153 (11)    <wsman:SelectorSet>
1154 (12)      <wsman:Selector Name="LUN"> 2 </wsman:Selector>
1155 (13)    </wsman:SelectorSet>
1156 (14)    <wsa:Action> http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
1157 (15)    </wsa:Action>
1158 (16)    <wsa:MessageID> urn:uuid:d9726315-bc91-430b-9ed8-ce5ffb858a91
1159 (17)    </wsa:MessageID>
1160 (18)  ...
1161 (19) </s:Header>
1162 (20) <s:Body> ... </s:Body>
1163 (21) </s:Envelope>

```

1164 The following definitions apply to the preceding message example:

- 1165 wsa:To
- 1166 the network (or transport-level) address of the service
- 1167 wsman:ResourceURI
- 1168 the ResourceURI of the resource class or resource instance to be accessed
- 1169 wsman:SelectorSet
- 1170 a wrapper for the selectors

1171 wsman:SelectorSet/wsman:Selector
 1172 identifies or selects the resource instance to be accessed, if more than one instance of the
 1173 resource exists
 1174 In this case, the selector is "LUN" (logical unit number), and the selected device is unit number
 1175 "2".

1176 wsa:Action
 1177 identifies which operation is to be carried out against the resource (in this case, a "Get")

1178 wsa:MessageID
 1179 identifies this specific message uniquely for tracking and correlation purposes
 1180 The format defined in [RFC 4122](#) is often used in the examples in this specification, but it is not
 1181 required.

1182 5.4.2.1 ResourceURI

1183 The ResourceURI is used to indicate the class resource or instance.

1184 **R5.4.2.1-1:** The format of the wsman:ResourceURI is unconstrained provided that it meets
 1185 [RFC 3986](#) requirements.

1186 The format and syntax of the ResourceURI is any valid URI according to [RFC 3986](#). Although there is
 1187 no default scheme, http: and urn: are common defaults. If http: is used, users may expect to find
 1188 Web-based documentation of the resource at that address. The wsa:To and the wsman:ResourceURI
 1189 elements work together to define the actual resource being targeted.

1190 **R5.4.2.1-2:** Vendor-specific or organization-specific URIs should contain the Internet domain
 1191 name in the first token sequence after the scheme, such as "example.org" in ResourceURI in the
 1192 following example.

1193 EXAMPLE:

```
1194 (20) <s:Header>
1195 (21)   <wsa:To> http://123.15.166.67/wsman </wsa:To>
1196 (22)   <wsman:ResourceURI>
1197 (23)     http://schemas.example.org/2005/02/hardware/physDisk
1198 (24)   </wsman:ResourceURI>
1199 (25)     ...
1200 (26) </s:Header>
```

1201 **R5.4.2.1-3:** When the default addressing model is used, the wsman:ResourceURI reference
 1202 parameter is required in messages with the following wsa:Action URIs:

1203 http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
 1204 http://schemas.xmlsoap.org/ws/2004/09/transfer/Put
 1205 http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
 1206 http://schemas.xmlsoap.org/ws/2004/09/transfer/Delete
 1207 http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
 1208 http://schemas.xmlsoap.org/ws/2004/09/enumeration/Pull
 1209 http://schemas.xmlsoap.org/ws/2004/09/enumeration/Renew
 1210 http://schemas.xmlsoap.org/ws/2004/09/enumeration/GetStatus
 1211 http://schemas.xmlsoap.org/ws/2004/09/enumeration/Release
 1212 http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe

- 1213 The following messages require the EPR to be returned in the SubscriptionManager element of the
 1214 SubscribeResponse message. The format of the EPR is determined by the service and might or
 1215 might not include the ResourceURI:
- 1216 `http://schemas.xmlsoap.org/ws/2004/08/eventing/Renew`
 1217 `http://schemas.xmlsoap.org/ws/2004/08/eventing/GetStatus`
- 1218 While the ResourceURI SOAP header is required when the WS-Management default addressing
 1219 mode is used, it may be short and of a very simple form, such as `http://example.com/*` or
 1220 `http://example.com/resource`.
- 1221 **R5.4.2.1-4:** For the request message of custom actions (methods), the ResourceURI header may
 1222 be present in the message to help route the message to the correct handler.
- 1223 **R5.4.2.1-5:** The ResourceURI element should not appear in other messages, such as responses
 1224 or events, unless the associated EPR includes it in its ReferenceParameters.
- 1225 In practice, the wsman:ResourceURI element is required only in requests to reference the targeted
 1226 resource class. Responses are not addressed to a management resource, so the
 1227 wsman:ResourceURI has no meaning in that context.
- 1228 **R5.4.2.1-6:** When the default addressing model is used and the wsman:ResourceURI element is
 1229 missing or in an incorrect form, the service shall issue a wsa:DestinationUnreachable fault with a
 1230 detail code of
- 1231 `http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/InvalidResourceURI`
- 1232 **R5.4.2.1-7:** The wsman:ResourceURI element shall be used to indicate only the identity of a
 1233 resource, and it may not be used to indicate the action being applied to that resource, which is
 1234 properly expressed using the wsa:Action URI.
- 1235 Custom WSDL-based methods have both a ResourceURI identity from the perspective of addressing
 1236 and a wsa:Action URI from the perspective of execution. In many cases, the ResourceURI is simply a
 1237 pseudonym for the WSDL identity and Port, and the wsa:Action URI is the specific method within that
 1238 port (or interface) definition.
- 1239 Although a single URI could theoretically be used alone to define an instance of a multi-instance
 1240 resource, it is recommended that the wsa:To element be used to locate the WS-Management service,
 1241 that the wsman:ResourceURI element be used to identify the resource class, and that the
 1242 wsman:SelectorSet element be used to reference the resource instance. If the resource consists of
 1243 only a single instance, then the wsman:ResourceURI element alone refers to the single instance.
- 1244 This usage is not a strict requirement, just a guideline. The service can use distinct selectors for any
 1245 given operation, even against the same resource class, and may allow or require selectors for the
 1246 Enumerate operation.
- 1247 See the recommendations in 7.2 regarding addressing uniformity.
- 1248 Custom actions have two distinct identities: the ResourceURI, which can identify the WSDL and port
 1249 (or interface), and the wsa:Action URI, which identifies the specific method. If only one method exists
 1250 in the interface, in a sense the ResourceURI and wsa:Action URI are identical.
- 1251 It is not an error to use the wsa:Action URI for the ResourceURI of a custom method, but both are still
 1252 required in the message for uniform processing on both clients and servers.

1253 EXAMPLE 1: The following action to reset a network card might have the following EPR usage:

```

1254 (1) <s:Header>
1255 (2)   <wsa:To>
1256 (3)     http://1.2.3.4/wsman/
1257 (4)   </wsa:To>
1258 (5)   <wsman:ResourceURI>http://example.org/2005/02/networkcards/reset
1259 (6)   </wsman:ResourceURI>
1260 (7)   <wsa:Action>
1261 (8)     http://example.org/2005/02/networkcards/reset
1262 (9)   </wsa:Action>
1263 (10)  ...
1264 (11) </s:Header>

```

1265 In many cases, the ResourceURI is equivalent to a WSDL name and port, and the wsa:Action URI
 1266 contains an additional token as a suffix, as in the following example.

1267 EXAMPLE 2:

```

1268 (1) <s:Header>
1269 (2)   <wsa:To>
1270 (3)     http://1.2.3.4/wsman
1271 (4)   </wsa:To>
1272 (5)   <wsman:ResourceURI>http://example.org/2005/02/networkcards
1273 (6)   </wsman:ResourceURI>
1274 (7)   <wsa:Action>
1275 (8)     http://example.org/2005/02/networkcards/reset
1276 (9)   </wsa:Action>
1277 (10)  ...
1278 (11) </s:Header>

```

1279 Finally, the ResourceURI may be completely unrelated to the wsa:Action URI, as in the following
 1280 example.

1281 EXAMPLE 3:

```

1282 (1) <s:Header>
1283 (2)   <wsa:To>http://1.2.3.4/wsman</wsa:To>
1284 (3)   <wsman:ResourceURI>
1285 (4)     http://example.org/products/management/networkcards
1286 (5)   </wsman:ResourceURI>
1287 (6)   <wsa:Action>
1288 (7)     http://example.org/2005/02/netcards/reset
1289 (8)   </wsa:Action>
1290 (9)   ...
1291 (10) </s:Header>

```

1292 All of these uses are legal.

1293 When used with subscriptions, the EPR described by wsa:Address and wsman:ResourceURI (and
 1294 optionally the wsman:SelectorSet values) identifies the event source to which the subscription is
 1295 directed. In many cases, the ResourceURI identifies a real or virtual event log, and the subscription is
 1296 intended to provide real-time notifications of any new entries added to the log. In many cases, the
 1297 wsman:SelectorSet element might not be used as part of the EPR.

1298 5.4.2.2 Selectors

1299 In the WS-Management default addressing model, selectors are optional elements used to identify
 1300 instances within a resource class. For operations such as Get or Put, the selectors are used to
 1301 identify a single instance of the resource class referenced by the ResourceURI.

1302 In practice, because the ResourceURI often acts as a table or a "class," the SelectorSet element is a
1303 discriminant used to identify a specific "row" or "instance." If only one instance of a resource class is
1304 implied by the ResourceURI, the SelectorSet can be omitted because the ResourceURI is acting as
1305 the full identity of the resource. If more than one selector value is required, the entire set of selectors
1306 is interpreted by the service in order to reference the specific instance. The selectors are interpreted
1307 as being separated by implied logical AND operators.

1308 In some information domains, the values referenced by the selectors are "keys" that are part of the
1309 resource content itself, whereas in other domains the selectors are part of a logical or physical
1310 directory system or search space. In these cases, the selectors are used to identify the resource, but
1311 are not part of the representation.

1312 **R5.4.2.2-1:** If a resource has more than one instance, a wsman:SelectorSet element may be
1313 used to distinguish which instance is targeted if the WS-Management default addressing model is
1314 in use. Any number of wsman:Selector values may appear with the wsman:SelectorSet element,
1315 as required to identify the precise instance of the resource class. The service may consider the
1316 case of selector names and values (see 13.6), as required by the underlying execution
1317 environment.

1318 If the client needs to discover the policy on how the case of selector values is interpreted, the service
1319 can provide metadata documents that describe this policy. The format of such metadata is beyond
1320 the scope of this specification.

1321 **R5.4.2.2-2:** All content within the SelectorSet element is to be treated as a single reference
1322 parameter with a scope relative to the ResourceURI.

1323 **R5.4.2.2-3:** A service using the WS-Management default addressing model shall examine all
1324 selectors in the message and process them as if they were logically joined by AND. If the set of
1325 selectors is incorrect for the targeted resource instance, a wsman:InvalidSelectors fault should be
1326 returned to the client with the following detail codes:

- 1327 • if selectors are missing:
1328 <http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/InsufficientSelectors>
- 1329 • if selector values are the wrong types:
1330 <http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/TypeMismatch>
- 1331 • if the selector value is of the correct type from the standpoint of XML types, but out of range
1332 or otherwise illegal in the specific information domain:
1333 <http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/InvalidValue>
- 1334 • if the name is not a recognized selector name
1335 <http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/UnexpectedSelectors>

1336 **R5.4.2.2-4:** The Selector Name attribute shall not be duplicated at the same level of nesting. If
1337 this occurs, the service should return a wsman:InvalidSelectors fault with the following detail
1338 code:

1339 <http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/DuplicateSelectors>

1340 This specification does not mandate the use of selectors. Some implementations may decide to use
1341 complex URI schemes in which the ResourceURI itself implicitly identifies the instance.

1342 The format of the SelectorSet element is as follows:

```

1343 (1) <s:Envelope
1344 (2)   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
1345 (3)   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
1346 (4)   xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
1347 (5)   <s:Header>
1348 (6)     ...
1349 (7)     <wsa:To> service transport address </wsa:To>
1350 (8)     <wsman:ResourceURI> ResourceURI </wsman:ResourceURI>
1351 (9)     <wsman:SelectorSet>
1352 (10)      <wsman:Selector Name="name"> value </wsman:Selector> +
1353 (11)      </wsman:SelectorSet> ?
1354 (12)     ...
1355 (13)   </s:Header>
1356 (14)   <s:Body> ... </s:Body>
1357 (15) </s:Envelope>

```

1358 The following definitions provide additional, normative constraints on the preceding outline:

1359 wsman:SelectorSet

1360 the wrapper for one or more Selector elements required to reference the instance

1361 wsman:SelectorSet/wsman:Selector

1362 used to describe the selector and its value

1363 If more than one selector is required, one Selector element exists for each part of the overall
 1364 selector. The value of this element is the Selector value.

1365 wsman:SelectorSet/wsman:Selector/@Name

1366 the name of the selector (to be treated in a case-insensitive manner)

1367 The value of a selector may be a nested EPR.

1368 EXAMPLE: In the following example, the selector on line 9 is a part of a SelectorSet that contains a nested
 1369 EPR (lines 10–18) with its own Address, ResourceURI, and SelectorSet elements:

```

1370 (1) <s:Envelope
1371 (2)   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
1372 (3)   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
1373 (4)   xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
1374 (5)   <s:Header>
1375 (6)     ...
1376 (7)     <wsman:SelectorSet>
1377 (8)       <wsman:Selector Name="Primary"> 123 </wsman:Selector>
1378 (9)       <wsman:Selector Name="EPR">
1379 (10)        <wsa:EndpointReference>
1380 (11)          <wsa:Address> address </wsa:Address>
1381 (12)          <wsa:ReferenceParameters>
1382 (13)            <wsman:ResourceURI> resource URI </wsman:ResourceURI>
1383 (14)            <wsman:SelectorSet>
1384 (15)              <wsman:Selector Name="name"> value </wsman:Selector>
1385 (16)            </wsman:SelectorSet>
1386 (17)          </wsa:ReferenceParameters>
1387 (18)        </wsa:EndpointReference>
1388 (19)      </wsman:Selector>
1389 (20)    </wsman:SelectorSet>
1390 (21)    ...
1391 (22)  </s:Header>
1392 (23)  <s:Body> ... </s:Body>
1393 (24) </s:Envelope>

```

1394 **R5.4.2.2-5:** For those services using the WS-Management default addressing model, the value of
 1395 a wsman:Selector shall be one of the following values:

- 1396 • a simple type as defined in the XML schema namespace
 1397 http://www.w3.org/2001/XMLSchema
- 1398 • a nested wsa:EndpointReference using the WS-Management default addressing model

1399 A service may fault selector usage with wsman:InvalidSelectors if the selector is not a simple type or
 1400 an EPR.

1401 **R5.4.2.2-6:** A conformant service may reject any selector or nested selector with a nested EPR
 1402 whose wsa:Address value is not the same as the primary wsa:To value or is not the Addressing
 1403 Anonymous URI.

1404 The primary purpose for this nesting mechanism is to allow resources that can answer questions
 1405 about other resources.

1406 **R5.4.2.2-7:** A service may fail to process a selector name of more than 2048 characters.

1407 **R5.4.2.2-8:** A service may fail to process a selector value of more than 4096 characters,
 1408 including any embedded selectors, and may fail to process a message that contains more than
 1409 8096 characters of content in the root SelectorSet element.

1410 **5.4.2.3 Faults for Default Addressing Model**

1411 When faults related to the information in the addressing model based on the default format are
 1412 generated, they may contain specific fault detail codes. These detail codes are called out separately
 1413 in 14.6 and do not apply when service-specific addressing is used.

1414 **5.4.3 Service-Specific Endpoint References**

1415 Although WS-Management specifies a default addressing model, in some cases this model is not
 1416 available or appropriate.

1417 **R5.4.3-1:** A conformant service may not understand the header values used by the
 1418 WS-Management default addressing model. If this is the case, and if the client marks the
 1419 wsman:ResourceURI with mustUnderstand="true", the service shall return an s:NotUnderstood
 1420 fault.

1421 **R5.4.3-2:** A conformant service may require additional header values to be present that are
 1422 beyond the scope of this specification.

1423 Services can thus use alternative addressing models for referencing resources with
 1424 WS-Management. These addressing models might or might not use ResourceURI or SelectorSet
 1425 elements and still be valid addressing models if they conform to the rules of Addressing.

1426 In addition to a defined alternative addressing model, a service might not explicitly define any
 1427 addressing model at all and instead use an opaque EPR generated at run-time, which is handled
 1428 according to the standard rules of Addressing.

1429 When such addressing models are used, the client application has to understand and interoperate
 1430 with discovery methods for acquiring EPRs that are beyond the scope of this specification.

1431 **5.4.4 mustUnderstand**

1432 This clause describes the use of the mustUnderstand attribute, regardless of whether an
 1433 implementation uses WS-Management Addressing (see 5.1) or the W3C Recommendation type of
 1434 WS-Addressing.

1435 The mustUnderstand attribute for SOAP headers is to be interpreted as a "must comply" instruction in
 1436 WS-Management. For example, if a SOAP header that is listed as being optional in this specification
 1437 is tagged with mustUnderstand="true", the service is required to comply or return a fault. To ensure
 1438 that the service treats a header as optional, the mustUnderstand attribute can be omitted.

1439 If the wsa:Action URI is not understood, the implementation might not know how to process the
 1440 message. So, for the following elements, the omission or inclusion of mustUnderstand="true" has no
 1441 real effect on the message in practice, because mustUnderstand is implied:

- 1442 • wsa:To
- 1443 • wsa:MessageID
- 1444 • wsa:RelatesTo
- 1445 • wsa:Action
- 1446 • wsa:ReplyTo
- 1447 • wsa:FaultTo

1448 **R5.4.4-1:** A conformant service shall process any of the preceding elements identically
 1449 regardless of whether mustUnderstand="true" is present.

1450 As a corollary, clients can omit mustUnderstand="true" from any of the preceding elements with no
 1451 change in meaning.

1452 **R5.4.4-2:** If a service cannot comply with a header marked with mustUnderstand="true", it
 1453 shall issue an s:NotUnderstood fault.

1454 The goal is for the service to be tolerant of inconsistent mustUnderstand usage by clients when the
 1455 request is not likely to be misinterpreted.

1456 It is important that clients using the WS-Management default addressing model (ResourceURI and
 1457 SelectorSet) use mustUnderstand="true" on the wsman:ResourceURI element to ensure that the
 1458 service is compliant with that addressing model. Implementations that use service-specific addressing
 1459 models will otherwise potentially ignore these header values and behave inconsistently with the
 1460 intentions of the client.

1461 5.4.5 wsa:To

1462 This clause describes the use of the Addressing wsa:To header regardless of whether an
 1463 implementation uses WS-Management Addressing (see 5.1) or the W3C Recommendation version of
 1464 WS-Addressing.

1465 In request messages, the wsa:To address contains the transport address of the service. In some
 1466 cases, this address is sufficient to locate the resource. In other cases, the service is a dispatching
 1467 agent for multiple resources. In these cases, the message typically contains additional headers to
 1468 allow the service to identify a resource within its scope. For example, when the default addressing
 1469 model is in use, these additional headers will be the ResourceURI and SelectorSet elements.

1470 NOTE: WS-Management does not preclude multiple listener services from coexisting on the same physical
 1471 system. Such services would be discovered and distinguished using mechanisms beyond the scope of this
 1472 specification.

1473 **R5.4.5-1:** The wsa:To header shall be present in all messages, whether requests, responses,
 1474 or events. In the absence of other requirements, it is recommended that the network address for
 1475 resources that require authentication be suffixed by the token sequence /wsman. If /wsman is
 1476 used, unauthenticated access should not be allowed.

1477 (1) <wsa:To> http://123.15.166.67/wsman </wsa:To>

1478 **R5.4.5-2:** In the absence of other requirements, it is recommended that the network address
 1479 for resources that do not require authentication be suffixed by the token sequence */wsman-anon*.
 1480 If */wsman-anon* is used, authenticated access shall not be required.

1481 (1) `<wsa:To> http://123.15.166.67/wsman-anon </wsa:To>`

1482 Including the network transport address in the SOAP message may seem redundant because the
 1483 network connection would already be established by the client. However, in cases where the
 1484 message is routed through intermediaries, the network transport address is required so that the
 1485 intermediaries can examine the message and make the connection to the actual endpoint.

1486 The *wsa:To* header may encompass any number of tokens required to locate the service and a group
 1487 of resources within that service.

1488 **R5.4.5-3:** The service should generate a fault when the *wsa:To* address cannot be processed
 1489 due to the following situations::

- 1490 • If the resource is offline, a *wsa:EndpointUnavailable* fault is returned with the following
 1491 detail code:
 1492 `http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/ResourceOffline`
- 1493 • If the resource cannot be located ("not found"), a *wsa:DestinationUnreachable* fault is
 1494 returned.
- 1495 • If the resource is valid, but internal errors occur, a *wsman:InternalError* fault is returned.
- 1496 • If the resource cannot be accessed for security reasons, a *wsman:AccessDenied* fault is
 1497 returned.

1498 **5.4.6 Other Addressing Headers**

1499 This clause describes the use of other Addressing headers, regardless of whether an implementation
 1500 uses WS-Management Addressing (see 5.1) or the W3C Recommendation version of WS-
 1501 Addressing.

1502 WS-Management depends on Addressing to describe the rules for use of other Addressing headers.

1503 **5.4.6.1 Processing Addressing Headers**

1504 The following additional addressing-related header blocks occur in WS-Management messages.

1505 **R5.4.6.1-1:** A conformant service shall recognize and process the following Addressing header
 1506 blocks.

- 1507 • **wsa:To**
- 1508 • **wsa:ReplyTo** (required when a response is expected)
- 1509 • **wsa:FaultTo** (optional)
- 1510 • **wsa:MessageID** (required)
- 1511 • **wsa:Action** (required)
- 1512 • **wsa:RelatesTo** (required in responses)

1513 The use of these header blocks is discussed in subsequent clauses.

1514 **5.4.6.2 wsa:ReplyTo**

1515 WS-Management requires the following usage of wsa:ReplyTo in addressing:

1516 **R5.4.6.2-1:** A wsa:ReplyTo header shall be present in all request messages when a reply is
 1517 required. This address shall be either a valid address for a new connection using any transport
 1518 supported by the service or the Addressing Anonymous URI, which indicates that the reply is to
 1519 be delivered over the same connection on which the request arrived. If the wsa:ReplyTo header
 1520 is missing, a wsa:MessageInformationHeaderRequired fault is returned.

1521 Some messages, such as event deliveries, SubscriptionEnd, and so on, do not require a response
 1522 and may omit a wsa:ReplyTo element.

1523 **R5.4.6.2-2:** A conformant service may require that all responses be delivered over the same
 1524 connection on which the request arrives. In this case, the URI discussed in R5.4.6.2-1 shall
 1525 indicate this. Otherwise, the service shall return a wsman:UnsupportedFeature fault with the
 1526 following detail code:

1527 <http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/AddressingMode>

1528 **R5.4.6.2-3:** When delivering events for which acknowledgement of delivery is required, the
 1529 sender of the event shall include a wsa:ReplyTo element and observe the usage in 10.8 of this
 1530 specification.

1531 **R5.4.6.2-4:** This rule intentionally left blank.

1532 **R5.4.6.2-5:** This rule intentionally left blank.

1533 Addressing allows clients to include client-defined reference parameters in wsa:ReplyTo headers.
 1534 Addressing requires that these reference parameters be extracted from requests and placed in the
 1535 responses by removing the ReferenceParameters wrapper and placing all of the values as top-level
 1536 SOAP headers in the response, as discussed in 5.1. This allows clients to better correlate responses
 1537 with the original requests. This step cannot be omitted.

1538 **EXAMPLE:** In the following example, the header x:someHeader is included in the reply message:

```

1539 (1) <s:Envelope
1540 (2)   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
1541 (3)   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
1542 (4)   xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
1543 (5)   <s:Header>
1544 (6)     ...
1545 (7)     <wsa:To> http://1.2.3.4/wsman </wsa:To>
1546 (8)     <wsa:ReplyTo>
1547 (9)       <wsa:Address>
1548 (10)        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
1549 (11)      </wsa:Address>
1550 (12)      <wsa:ReferenceParameters>
1551 (13)        <x:someHeader xmlns:x="..."> user-defined content </x:someHeader>
1552 (14)      </wsa:ReferenceParameters>
1553 (15)    </wsa:ReplyTo>
1554 (16)    ...
1555 (17)  </s:Header>
1556 (18)  <s:Body> ... </s:Body>
1557 (19) </s:Envelope>

```

1558 **R5.4.6.2-6:** If the wsa:ReplyTo address is not usable or is missing, the service should not reply to
 1559 the request and it should close or terminate the connection according to the rules of the current
 1560 network transport. In these cases, the service should locally log some type of entry to help locate
 1561 the client defect later.

1562 **5.4.6.3 wsa:FaultTo**

1563 WS-Management qualifies the use of wsa:FaultTo as indicated in this clause.

1564 **R5.4.6.3-1:** A conformant service may support a wsa:FaultTo address that is distinct from the
 1565 wsa:ReplyTo address. If such a request is made and is not supported by the service, a
 1566 wsman:UnsupportedFeature fault shall be returned with the following detail code:

1567 `http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/AddressingMode`

1568 If both the wsa:FaultTo and wsa:ReplyTo headers are omitted from a request, transport-level
 1569 mechanisms are typically used to fail the request because the address to which the fault is to be sent
 1570 is uncertain. In such a case, it is not an error for the service to simply shut down the connection.

1571 **R5.4.6.3-2:** If wsa:FaultTo is omitted, the service shall return the fault to the wsa:ReplyTo
 1572 address if a fault occurs.

1573 **R5.4.6.3-3:** A conformant service may require that all faults be delivered to the client over the
 1574 same transport or connection on which the request arrives. In this case, the URI shall be the
 1575 Addressing Anonymous URI. If services do not support separately addressed fault delivery and
 1576 the wsa:FaultTo is any other address, a wsman:UnsupportedFeature fault shall be returned with
 1577 the following detail code:

1578 `http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/AddressingMode`

1579 NOTE: This specification does not restrict richer implementations from fully supporting wsa:FaultTo.

1580 **R5.4.6.3-4:** This rule intentionally left blank.

1581 EXAMPLE: In the following example, the header x:someHeader is included in fault messages if they occur:

```

1582 (1) <s:Envelope
1583 (2)   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
1584 (3)   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
1585 (4)   xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
1586 (5)   <s:Header>
1587 (6)     ...
1588 (7)     <wsa:To> http://1.2.3.4/wsman </wsa:To>
1589 (8)     <wsa:FaultTo>
1590 (9)       <wsa:Address>
1591 (10)        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
1592 (11)      </wsa:Address>
1593 (12)      <wsa:ReferenceParameters>
1594 (13)        <x:someHeader xmlns:x="..."> user-defined content </x:someHeader>
1595 (14)      </wsa:ReferenceParameters>
1596 (15)    </wsa:FaultTo>
1597 (16)    ...
1598 (17)  </s:Header>
1599 (18)  <s:Body> ... </s:Body>
1600 (19) </s:Envelope>

```

1601 **R5.4.6.3-5:** If the wsa:FaultTo address is not usable, the service should not reply to the request.
 1602 Similarly, if according to WS-Addressing processing rules there is no suitable address to send a
 1603 fault to, it should not reply and should close the network connection. In these cases, the service
 1604 should locally log some type of entry to help locate the client defect later.

1605 **R5.4.6.3-6:** The service shall properly duplicate the wsa:Address of the wsa:FaultTo element in
 1606 the wsa:To of the reply, even if some of the information is not understood by the service.

1607 This rule applies in cases where the client includes private content suffixes on the HTTP or HTTPS
 1608 address that the service does not understand. If the service removes this information when
 1609 constructing the address, the subsequent message might not be correctly processed.

1610 **5.4.6.4 wsa:MessageID and wsa:RelatesTo**

1611 WS-Management qualifies the use of wsa:MessageID and wsa:RelatesTo as follows:

1612 **R5.4.6.4-1:** The MessageID and RelatesTo URIs may be of any format, as long as they are valid
 1613 URIs according to [RFC 3986](#). Two URIs are considered different even if the characters in the
 1614 URIs differ only by case.

1615 The following two formats are endorsed by this specification. The first is considered a best
 1616 practice because it is backed by [RFC 4122](#):

1617 urn:uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
 1618 or
 1619 uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

1620 In these formats, each x is an uppercase or lowercase hexadecimal digit (lowercase is required
 1621 by [RFC 4122](#)); there are no spaces or other tokens. The value may be a DCE-style universally
 1622 unique identifier (UUID) with provable uniqueness properties in this format, however, it is not
 1623 necessary to have provable uniqueness properties in the URIs used in the wsa:MessageID and
 1624 wsa:RelatesTo headers.

1625 Regardless of format, the URI should not exceed the maximum defined in R13.1-6.

1626 UUIDs have a numeric meaning as well as a string meaning, and this can lead to confusion. A UUID
 1627 in lowercase is a different URI from the same UUID in uppercase. This is because URIs are case-
 1628 sensitive. If a UUID is converted to its decimal equivalent the case of the original characters is lost.
 1629 WS-Management works with the URI value itself, not the underlying decimal equivalent
 1630 representation. Services are free to *interpret* the URI in any way, but are not allowed to alter the case
 1631 usage when repeating the message or any of the MessageID values in subsequent messages.

1632 The [RFC 4122](#) requires the digits to be lowercase, which is the responsibility of the client. The service
 1633 simply processes the values as URI values and is not required to analyze the URI for correctness or
 1634 compliance. The service replicates the client usage in the wsa:RelatesTo reply header and is not
 1635 allowed to alter the case usage.

1636 **R5.4.6.4-2:** The MessageID should be generated according to any algorithm that ensures that no
 1637 two MessageIDs are repeated. Because the value is treated as case-sensitive (R5.4.6.4-1),
 1638 confusion can arise if the same value is reused differing only in case. As a result, the service shall
 1639 not create or employ MessageID values that differ only in case. For any message transmitted by
 1640 the service, the MessageID shall not be reused.

1641 The client ensures that MessageID values are not reused in requests. Although services and clients
 1642 can issue different MessageIDs that differ only in case, the service is not required to detect this
 1643 difference, nor is it required to analyze the URI for syntactic correctness or repeated use.

1644 **R5.4.6.4-3:** The RelatesTo element shall be present in all response messages and faults, shall
 1645 contain the MessageID of the associated request message, and shall match the original in case,
 1646 being treated as a URI value and not as a binary UUID value.

1647 **R5.4.6.4-4:** If the MessageID is not parsable or is missing, a
 1648 wsa:InvalidMessageInformationHeader fault should be returned.

1649 EXAMPLE: The following examples show wsa:MessageID usage:

```

1650 (20) <wsa:MessageID>
1651 (21)     uuid:d9726315-bc91-430b-9ed8-ce5ffb858a91
1652 (22) </wsa:MessageID>
1653 (23)
1654 (24) <wsa:MessageID>
1655 (25)     anotherScheme:ID/12310/1231/16607/25
1656 (26) </wsa:MessageID>

```

1657 5.4.6.5 wsa:Action

1658 The wsa:Action URI indicates the "operation" being invoked against the resource.

1659 **R5.4.6.5-1:** The wsa:Action URI shall not be used to identify the specific resource class or
 1660 instance, but only to identify the operation to use against that resource.

1661 **R5.4.6.5-2:** For all resource endpoints, a service shall return a wsa:ActionNotSupported fault if a
 1662 requested action is not supported by the service for the specified resource.

1663 In other words, to model the "Get" of item "Disk", the wsa:Action URI contains the "Get". The wsa:To,
 1664 and potentially other SOAP headers, indicate *what* is being accessed. When the default addressing
 1665 model is used, for example, the ResourceURI typically contains the reference to the "Disk" and the
 1666 SelectorSet identifies which disk. Other service-specific addressing models can factor the identity of
 1667 the resource in different ways.

1668 Implementations are free to support additional custom methods that combine the notion of "Get" and
 1669 "Disk" into a single "GetDisk" action if they strive to support the separated form to maximize
 1670 interoperation. One of the main points behind WS-Management is to unify common methods
 1671 wherever possible.

1672 **R5.4.6.5-3:** If a service exposes any of the following types of capabilities, a conformant service
 1673 shall at least expose that capability using the definitions in Table 4 according to the rules of this
 1674 specification. The service may optionally expose additional similar functionality using a distinct
 1675 wsa:Action URI.

1676 **Table 4 – wsa:Action URI Descriptions**

Action URI	Description
http://schemas.xmlsoap.org/ws/2004/09/transfer/Get	Models any simple single item retrieval
http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse	Response to "Get"
http://schemas.xmlsoap.org/ws/2004/09/transfer/Put	Models an update of an entire item
http://schemas.xmlsoap.org/ws/2004/09/transfer/PutResponse	Response to "Put"
http://schemas.xmlsoap.org/ws/2004/09/transfer/Create	Models creation of a new item
http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse	Response to "Create"
http://schemas.xmlsoap.org/ws/2004/09/transfer/Delete	Models the deletion of an item
http://schemas.xmlsoap.org/ws/2004/09/transfer/DeleteResponse	Response to "Delete"
http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate	Begins an enumeration or query
http://schemas.xmlsoap.org/ws/2004/09/enumeration/EnumerateResponse	Response to "Enumerate"
http://schemas.xmlsoap.org/ws/2004/09/enumeration/Pull	Retrieves the next batch of results from enumeration
http://schemas.xmlsoap.org/ws/2004/09/enumeration/PullResponse	Response to "Pull"

Action URI	Description
http://schemas.xmlsoap.org/ws/2004/09/enumeration/Renew	Renews an enumerator that may have timed out (not required in WS-Management)
http://schemas.xmlsoap.org/ws/2004/09/enumeration/RenewResponse	Response to "Renew" (not required in WS-Management)
http://schemas.xmlsoap.org/ws/2004/09/enumeration/GetStatus	Gets the status of the enumerator (not required in WS-Management)
http://schemas.xmlsoap.org/ws/2004/09/enumeration/GetStatusResponse	Response to "GetStatus" (not required in WS-Management)
http://schemas.xmlsoap.org/ws/2004/09/enumeration/Release	Releases an active enumerator
http://schemas.xmlsoap.org/ws/2004/09/enumeration/ReleaseResponse	Response to "Release"
http://schemas.xmlsoap.org/ws/2004/09/enumeration/EnumerationEnd	Notifies that an enumerator has terminated (not required in WS-Management)
http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe	Models a subscription to an event source
http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscribeResponse	Response to "Subscribe"
http://schemas.xmlsoap.org/ws/2004/08/eventing/Renew	Renews a subscription prior to its expiration
http://schemas.xmlsoap.org/ws/2004/08/eventing/RenewResponse	Response to "Renew"
http://schemas.xmlsoap.org/ws/2004/08/eventing/GetStatus	Requests the status of a subscription
http://schemas.xmlsoap.org/ws/2004/08/eventing/GetStatusResponse	Response to "GetStatus"
http://schemas.xmlsoap.org/ws/2004/08/eventing/Unsubscribe	Removes an active subscription
http://schemas.xmlsoap.org/ws/2004/08/eventing/UnsubscribeResponse	Response to "Unsubscribe"
http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscriptionEnd	Delivers a message to indicate that a subscription has terminated
http://schemas.dmtf.org/wbem/wsman/1/wsman/Events	Delivers batched events based on a subscription
http://schemas.dmtf.org/wbem/wsman/1/wsman/Heartbeat	A pseudo-event that models a heartbeat of an active subscription; delivered when no real events are available, but used to indicate that the event subscription and delivery mechanism is still active
http://schemas.dmtf.org/wbem/wsman/1/wsman/DroppedEvents	A pseudo-event that indicates that the real event was dropped
http://schemas.dmtf.org/wbem/wsman/1/wsman/Ack	Used by event subscribers to acknowledge receipt of events; allows event streams to be strictly sequenced
http://schemas.dmtf.org/wbem/wsman/1/wsman/Event	Used for a singleton event that does not define its own action

1677 **R5.4.6.5-4:** A custom action may be supported if the operation is a custom method whose
1678 semantic meaning is not present in the table.

1679 **R5.4.6.5-5:** All notifications shall contain a unique action URI that identifies the type of the event
1680 delivery. For singleton notifications with only one event per message (the delivery mode
1681 <http://schemas.xmlsoap.org/ws/2004/08/eventing/DeliveryModes/Push>), the wsa:Action URI

1682 defines the event type. For other delivery modes, the Action varies, as described in clause 10.2.7
1683 of this specification.

1684 5.4.6.6 wsa:From

1685 The wsa:From header can be used in any messages, responses, or events to indicate the source.
1686 When the same connection is used for both request and reply, this header provides no useful
1687 information, but can be useful in cases where the response arrives on a different connection.

1688 **R5.4.6.6-1:** A conformant service may include a wsa:From address in the message. A
1689 conformant service should process any incoming message that has a wsa:From element.

1690 **R5.4.6.6-2:** A conformant service should not fault any message with a wsa:From element,
1691 regardless of whether the mustUnderstand attribute is included.

1692 NOTE: Processing the wsa:From header is trivial because it has no effect on the meaning of the
1693 message. The *From* address is primarily for auditing and logging purposes.

1694 6 WS-Management Control Headers

1695 WS-Management defines several SOAP headers that can be used with any operation.

1696 6.1 wsman:OperationTimeout

1697 Most management operations are time-critical due to quality-of-service constraints and obligations. If
1698 operations cannot be completed in a specified time, the service returns a fault so that a client can
1699 comply with its obligations. The following header value can be supplied with any WS-Management
1700 message to indicate that the client expects a response or a fault within the specified time:

1701 (1) `<wsman:OperationTimeout> xs:duration </wsman:OperationTimeout>`

1702 **R6.1-1:** All request messages may contain a wsman:OperationTimeout header element that
1703 indicates the maximum amount of time the client is willing to wait for the service to issue a
1704 response. The service should interpret the timeout countdown as beginning from the point the
1705 message is processed until a response is generated.

1706 **R6.1-2:** The service should *immediately* issue a wsman:TimedOut fault if the countdown time is
1707 exceeded and the operation is not yet complete. If the OperationTimeout value is not valid, a
1708 wsa:InvalidMessageInformationHeader fault should be returned.

1709 **R6.1-3:** If the service does not support user-defined timeouts, a wsman:UnsupportedFeature
1710 fault should be returned with the following detail code:

1711 `http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/OperationTimeout`

1712 **R6.1-4:** If the wsman:OperationTimeout element is omitted, the service may interpret this
1713 omission as an instruction to block indefinitely until a response is available, or it may impose a
1714 default timeout.

1715 These rules do not preclude services from supporting infinite or very long timeouts. Because network
1716 connections seldom block indefinitely with no traffic occurring, some type of transport timeout is likely.
1717 Also the countdown is initiated from the time the message is received, so network latency is not
1718 included. If a client needs to discover the range of valid timeouts or defaults, metadata can be
1719 retrieved, but the format of such metadata is beyond the scope of this specification.

1720 If the timeout occurs in such a manner that the service has already performed some of the work
1721 associated with the request, the service state reaches an anomalous condition. This specification
1722 does not attempt to address behavior in this situation. Clearly, services can attempt to undo the

1723 effects of any partially complete operations, but this is not always practical. In such cases, the service
1724 can keep a local log of requests and operations, which the client can query later.

1725 For example, if a Delete operation is in progress and a timeout occurs, the service decides whether to
1726 attempt a rollback or roll-forward of the deletion, even though it issues a wsman:TimedOut fault. The
1727 service can elect to include additional information in the fault (see 14.5) regarding its internal policy in
1728 this regard. The service can attempt to return to the state that existed before the operation was
1729 attempted, but this is not always possible.

1730 **R6.1-5:** If the mustUnderstand attribute is applied to the wsman:OperationTimeout element and
1731 the service understands wsman:OperationTimeout, the service shall observe the requested value
1732 or return the fault specified in R6.1-2. The service should attempt to complete the request within
1733 the specified time or issue a fault without any further delay.

1734 Clients can always omit the mustUnderstand header for uniform behavior against all implementations.
1735 It is not an error for a compliant service to ignore the timeout value or treat it as a hint if
1736 mustUnderstand is omitted.

1737 EXAMPLE: The following is an example of a correctly formatted 30-second timeout in the SOAP header:

1738 (1) `<wsman:OperationTimeout>PT30S</wsman:OperationTimeout>`

1739 If the transport timeout occurs before the actual wsman:OperationTimeout, the operation can be
1740 treated as specified in 13.3, the same as a failed connection. In practice, the network transport
1741 timeout can be configured to be longer than any expected wsman:OperationTimeout.

1742 6.2 wsman:MaxEnvelopeSize

1743 To prevent a response beyond the capability of the client, the request message can contain a
1744 restriction on the response size.

1745 The following header value may be supplied with any WS-Management message to indicate that the
1746 client expects a response whose total SOAP envelope does not exceed the specified number of
1747 octets:

1748 (1) `<wsman:MaxEnvelopeSize> xs:positiveInteger </wsman:MaxEnvelopeSize>`

1749 The limitation is on the entire envelope. Resource-constrained implementations need a reliable figure
1750 for the required amount of memory for all SOAP processing, not just the SOAP Body.

1751 **R6.2-1:** All request messages may contain a wsman:MaxEnvelopeSize header element that
1752 indicates the maximum number of octets (not characters) in the entire SOAP envelope in the
1753 response. If the service cannot compose a reply within the requested size, it should return a
1754 wsman:EncodingLimit fault with the following detail code:

1755 `http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/MaxEnvelopeSize`

1756 **R6.2-2:** If the mustUnderstand attribute is set to "true", the service shall comply with the
1757 request. If the response would exceed the maximum size, the service should return a
1758 wsman:EncodingLimit fault. Because a service might execute the operation prior to knowing the
1759 response size, the service should undo any effects of the operation before issuing the fault. If the
1760 operation cannot be reversed (such as a destructive Put or Delete, or a Create), the service shall
1761 indicate that the operation succeeded in the wsman:EncodingLimit fault with the following detail
1762 code:

1763 `http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/UnreportableSuccess`

1764 **R6.2-3:** If the mustUnderstand attribute is set to "false", the service may ignore the header.

1765 **R6.2-4:** Services should reject any MaxEnvelopeSize value less than 8192 octets. This number
 1766 is the safe minimum in which faults can be reliably encoded for all character sets. If the requested
 1767 size is less than this, the service should return a wsman:EncodingLimit fault with the following
 1768 detail code:

1769 `http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/MinimumEnvelopeLimit`

1770 A service might have its own encoding limit independent of what the client specifies, and the same
 1771 fault applies.

1772 **R6.2-5:** If the service cannot compose a reply within its own internal limits, the service should
 1773 return a wsman:EncodingLimit fault with the following detail code:

1774 `http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/ServiceEnvelopeLimit`

1775 The definition of the wsman:MaxEnvelopeSize element in the schema contains a Policy attribute
 1776 because this element is used for other purposes. This specification does not define a meaning for the
 1777 Policy attribute when the wsman:MaxEnvelopeSize element is used as a SOAP header.

1778 **R6.2-6:** Clients should not add the Policy attribute to the wsman:MaxEnvelopeSize element
 1779 when it is used as a SOAP header. Services should ignore the Policy attribute if it appears in the
 1780 wsman:MaxEnvelopeSize element when used as a SOAP header.

1781 **6.3 wsman:Locale**

1782 Management operations often span locales, and many items in responses can require translation.
 1783 Typically, translation is required for descriptive information, intended for human readers, that is sent
 1784 back in the response. If the client requires such output to be translated into a specific language, it can
 1785 employ the optional wsman:Locale header, which makes use of the standard XML attribute xml:lang,
 1786 as follows:

1787 `(1) <wsman:Locale xml:lang="xs:language" s:mustUnderstand="false"/>`

1788 **R6.3-1:** If the mustUnderstand attribute is omitted or set to "false", the service should use this
 1789 value when composing the response message and adjust any localizable values accordingly.
 1790 This use is recommended for most cases. The locale is treated as a hint in this case.

1791 **R6.3-2:** If the mustUnderstand attribute is set to "true", the service shall ensure that the replies
 1792 contain localized information where appropriate, or else the service shall issue a
 1793 wsman:UnsupportedFeature fault with the following detail code:

1794 `http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDetail/Locale`

1795 A service may always fault if wsman:Locale contains s:mustUnderstand set to "true", because it
 1796 may not be able to ensure that the reply is localized.

1797 Some implementations delegate the request to another subsystem for processing, so the service
 1798 cannot be certain that the localization actually occurred.

1799 **R6.3-3:** The value of the xml:lang attribute in the wsman:Locale header shall be a valid
 1800 [RFC 5646](#) language code.

1801 **R6.3-4:** In any response, event, or singleton message, the service should include the xml:lang
 1802 attribute in the s:Envelope (or other elements) to signal to the receiver that localized content
 1803 appears in the body of the message. This attribute may be omitted if no descriptive content
 1804 appears in the body. Including the xml:lang attribute is not an error, even if no descriptive content
 1805 occurs.