



Document Number: DSP0200

Date: 2013-05-07

Version: 1.4.0a

## CIM Operations over HTTP

### Information for Work-in-Progress version:

**IMPORTANT:** This document is not a standard. It does not necessarily reflect the views of the DMTF or all of its members. Because this document is a Work in Progress, it may still change, perhaps profoundly. This document is available for public review and comment until the stated expiration date.

**It expires on: 2013-10-31**

**Provide any comments through the DMTF Feedback Portal:**  
<http://www.dmtf.org/standards/feedback>

Document Type: Specification

Document Status: Work in Progress

Document Language: en-US

10 Copyright Notice

11 Copyright © 1999-2013 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26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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	Introduction.....	8
36	Requirements .....	8
37	1    Scope .....	11
38	2    Normative References.....	11
39	3    Terms and Definitions .....	12
40	4    Abbreviated Terms and Document Conventions .....	14
41	4.1    Abbreviated Terms.....	14
42	4.2    Document Conventions.....	14
43	5    CIM-XML Message Syntax and Semantics.....	14
44	5.1    Well-Formed, Valid, and Loosely Valid Documents .....	15
45	5.2    Operational Semantics.....	15
46	5.3    Operation Correlators .....	17
47	5.3.1    Overview .....	17
48	5.3.2    Representation.....	17
49	5.3.3    Implementation Requirements and Compatibility for Operation Messages .....	17
50	5.3.4    Implementation Requirements and Compatibility for Export Messages .....	18
51	5.4    CIM Operation Syntax and Semantics.....	18
52	5.4.1    Method Invocations.....	18
53	5.4.1.1    Simple Operations.....	19
54	5.4.1.2    Multiple Operations .....	19
55	5.4.1.3    Status Codes.....	20
56	5.4.2    Intrinsic Methods.....	22
57	5.4.2.1    GetClass.....	23
58	5.4.2.2    GetInstance .....	24
59	5.4.2.3    DeleteClass .....	26
60	5.4.2.4    DeleteInstance .....	26
61	5.4.2.5    CreateClass.....	27
62	5.4.2.6    CreateInstance .....	28
63	5.4.2.7    ModifyClass.....	29
64	5.4.2.8    ModifyInstance .....	31
65	5.4.2.9    EnumerateClasses .....	33
66	5.4.2.10    EnumerateClassNames .....	34
67	5.4.2.11    EnumerateInstances (DEPRECATED) .....	34
68	5.4.2.12    EnumerateInstanceNames (DEPRECATED) .....	36
69	5.4.2.13    ExecQuery (DEPRECATED) .....	37
70	5.4.2.14    Associators (PARTLY DEPRECATED).....	38
71	5.4.2.15    AssociatorNames (PARTLY DEPRECATED).....	39
72	5.4.2.16    References (PARTLY DEPRECATED).....	40
73	5.4.2.17    ReferenceNames (PARTLY DEPRECATED).....	42
74	5.4.2.18    GetProperty (DEPRECATED).....	43
75	5.4.2.19    SetProperty (DEPRECATED) .....	43
76	5.4.2.20    GetQualifier .....	44
77	5.4.2.21    SetQualifier.....	45
78	5.4.2.22    DeleteQualifier.....	45
79	5.4.2.23    EnumerateQualifiers .....	46
80	5.4.2.24    Pulled Enumeration Operations .....	46
81	5.4.3    Namespace Manipulation Using the CIM_Namespace Class .....	66
82	5.4.3.1    Namespace Creation .....	66
83	5.4.3.2    Namespace Deletion.....	67
84	5.4.3.3    Manipulation and Query of Namespace Information.....	67
85	5.4.3.4    Use of the __Namespace Pseudo Class (DEPRECATED) .....	67

86	5.4.4	Functional Profiles .....	68
87	5.4.5	Extrinsic Method Invocation .....	69
88	5.5	CIM Export Syntax and Semantics .....	70
89	5.5.1	Export Method Invocations .....	70
90	5.5.1.1	Simple Export .....	71
91	5.5.1.2	Multiple Export .....	71
92	5.5.1.3	Status Codes .....	71
93	5.5.2	Export Methods .....	72
94	5.5.2.1	ExportIndication .....	74
95	5.5.3	Functional Profiles .....	74
96	6	Encapsulation of CIM-XML Messages .....	75
97	6.1	WBEM clients, WBEM servers, and WBEM listeners .....	75
98	6.2	Use of M-POST .....	76
99	6.2.1	Use of the Ext Header .....	76
100	6.2.2	Naming of Extension Headers .....	76
101	6.3	Extension Headers Defined for CIM-XML Message Requests and Responses .....	77
102	6.3.1	Encoding of CIM Element Names within HTTP Headers and Trailers .....	77
103	6.3.2	Encoding of CIM Object Paths within HTTP Headers and Trailers .....	78
104	6.3.3	CIMOperation .....	79
105	6.3.4	CIMExport .....	80
106	6.3.5	CIMProtocolVersion .....	80
107	6.3.6	CIMMethod .....	81
108	6.3.7	CIMObject .....	81
109	6.3.8	CIMExportMethod .....	82
110	6.3.9	CIMBatch .....	83
111	6.3.10	CIMExportBatch .....	84
112	6.3.11	CIMError .....	84
113	6.3.12	CIMRoleAuthenticate .....	85
114	6.3.13	CIMRoleAuthorization .....	85
115	6.3.14	CIMStatusCodeDescription .....	86
116	6.3.15	WBEMServerResponseTime .....	86
117	7	HTTP Requirements and Usage .....	86
118	7.1	HTTP and HTTPS Support .....	86
119	7.2	Use of Standard HTTP Headers .....	87
120	7.2.1	Accept .....	87
121	7.2.2	Accept-Charset .....	87
122	7.2.3	Accept-Encoding .....	87
123	7.2.4	Accept-Language .....	88
124	7.2.5	Accept-Ranges .....	88
125	7.2.6	Allow .....	88
126	7.2.7	Authorization .....	88
127	7.2.8	Cache-Control .....	88
128	7.2.9	Connection .....	89
129	7.2.10	Content-Encoding .....	89
130	7.2.11	Content-Language .....	89
131	7.2.12	Content-Range .....	89
132	7.2.13	Content-Type .....	90
133	7.2.14	Expires .....	90
134	7.2.15	If-Range .....	90
135	7.2.16	Proxy-Authenticate .....	90
136	7.2.17	Range .....	90
137	7.2.18	WWW-Authenticate .....	90
138	7.3	Errors and Status Codes .....	90
139	7.4	Security Considerations .....	92
140	7.4.1	Authentication .....	92
141	7.4.2	Message Encryption .....	93

142	7.5	Determining WBEM server Capabilities.....	93
143	7.5.1	Determining WBEM server Capabilities through CIM Classes.....	94
144	7.5.2	Determining WBEM server Capabilities through the HTTP Options .....	95
145	7.5.2.1	CIMSupportedFunctionalGroups.....	96
146	7.5.2.2	CIMSupportsMultipleOperations .....	97
147	7.5.2.3	CIMSupportedQueryLanguages (DEPRECATED) .....	97
148	7.5.2.4	CIMValidation .....	97
149	7.6	Other HTTP Methods.....	98
150	7.7	Discovery and Addressing .....	98
151	7.8	Internationalization Considerations.....	99
152	ANNEX A (Informative)	Examples of Message Exchanges.....	100
153	A.1	Retrieval of a Single Class Definition.....	100
154	A.2	Retrieval of a Single Instance Definition .....	101
155	A.3	Deletion of a Single Class Definition.....	102
156	A.4	Deletion of a Single Instance Definition .....	103
157	A.5	Creation of a Single Class Definition .....	104
158	A.6	Creation of a Single Instance Definition.....	105
159	A.7	Enumeration of Class Names .....	106
160	A.8	Enumeration of Instances .....	107
161	A.9	Retrieval of a Single Property .....	108
162	A.10	Execution of an Extrinsic Method .....	110
163	A.11	Indication Delivery Example .....	111
164	A.12	Subscription Example .....	112
165	A.13	Multiple Operations Example.....	119
166	ANNEX B (informative)	LocalOnly Parameter Discussion.....	122
167	B.1	Explanation of the Deprecated 1.1 Interpretation .....	122
168	B.2	Risks of Using the 1.1 Interpretation.....	123
169	B.3	Techniques for Differentiating between the 1.0 Interpretation and 1.1 Interpretation .....	124
170	ANNEX C (normative)	Generic Operations Mapping.....	125
171	C.1	Operations .....	125
172	C.1.1	GetInstance.....	126
173	C.1.2	DeleteInstance.....	127
174	C.1.3	ModifyInstance.....	128
175	C.1.4	CreateInstance.....	128
176	C.1.5	GetClassInstancesWithPath .....	129
177	C.1.6	GetClassInstancePaths .....	130
178	C.1.7	GetAssociatedInstancesWithPath .....	131
179	C.1.8	GetAssociatedInstancePaths.....	132
180	C.1.9	GetReferencingInstancesWithPath.....	132
181	C.1.10	GetReferencingInstancePaths.....	134
182	C.1.11	OpenClassInstancesWithPath .....	135
183	C.1.12	OpenClassInstancePaths .....	136
184	C.1.13	OpenAssociatedInstancesWithPath .....	137
185	C.1.14	OpenAssociatedInstancePaths.....	138
186	C.1.15	OpenReferencingInstancesWithPath.....	139
187	C.1.16	OpenReferencingInstancePaths.....	140
188	C.1.17	OpenQueryInstances .....	141
189	C.1.18	PullInstancesWithPath .....	142
190	C.1.19	PullInstancePaths .....	142
191	C.1.20	PullInstances.....	143
192	C.1.21	CloseEnumeration .....	144
193	C.1.22	EnumerationCount .....	144
194	C.1.23	InvokeMethod .....	144
195	C.1.24	InvokeStaticMethod .....	145
196	C.1.25	GetClass .....	146

197	C.1.26 DeleteClass.....	147
198	C.1.27 ModifyClass .....	147
199	C.1.28 CreateClass .....	148
200	C.1.29 GetTopClassesWithPath .....	148
201	C.1.30 GetTopClassPaths.....	149
202	C.1.31 GetSubClassesWithPath .....	150
203	C.1.32 GetSubClassPaths.....	151
204	C.1.33 GetAssociatedClassesWithPath .....	151
205	C.1.34 GetAssociatedClassPaths .....	152
206	C.1.35 GetReferencingClassesWithPath .....	153
207	C.1.36 GetReferencingClassPaths .....	154
208	C.1.37 GetQualifierType.....	155
209	C.1.38 DeleteQualifierType .....	155
210	C.1.39 ModifyQualifierType .....	156
211	C.1.40 CreateQualifierType.....	156
212	C.1.41 EnumerateQualifierTypesWithPath .....	157
213	ANNEX D (informative) Change Log .....	159
214	Bibliography .....	161
215		

216 **Tables**

217	Table 1 – Status Codes Returned by an <Error> Child element .....	21
218	Table 2 – Mapping of Intrinsic Method Pseudo-Types to XML Elements.....	23
219	Table 3 – Root-Directed Tree of Functional Profile Dependencies .....	69
220	Table 4 – Symbolic Names for Referencing Error Codes.....	72
221	Table 5 – Mapping of Export Method Pseudo-Types to XML Elements.....	74
222	Table 6 – Functional Groups of Export Methods .....	75
223	Table 7 – Comparison of Properties Returned by GetInstance in Versions 1.0 and 1.1 .....	123
224	Table 8 – Comparison of Properties Returned by a Call to GetInstance in Versions 1.0 and 1.1.....	124
225	Table 9 – Mapping of generic operations to CIM-XML operations .....	125
226		

227

## Foreword

228 CIM Operations over HTTP (DSP0200) was prepared by the DMTF CIM-XML Working Group.

229

## Introduction

230 This document defines a mapping of CIM-XML messages to the Hypertext Transfer Protocol (HTTP and  
231 HTTPS) so that implementations of CIM can operate in an open, standardized manner. It also defines the  
232 notion of *conformance* in the context of this mapping, and it describes the behavior an implementation of  
233 CIM shall exhibit to be a conforming CIM implementation.

234 Unless otherwise noted, the term HTTP is used in this document to mean both HTTP and HTTPS.

235 This document is structured as follows:

- 236 • [Clause 5](#) describes the CIM-XML messages that form the HTTP payload using XML. It specifies  
237 the syntax and semantics of the message requests and their corresponding responses.
- 238 • [Clause 6](#) describes the encapsulation of these messages in HTTP request and response  
239 messages, with examples of each. It also describes the extension headers used to convey  
240 additional CIM-specific semantics in the HTTP Header.
- 241 • [Clause 7](#) presents details of other aspects of the encapsulation:
  - 242 – HTTP version support
  - 243 – Use of standard HTTP headers
  - 244 – HTTP error codes
  - 245 – Security considerations

## 246 Requirements

247 There are many different ways CIM-XML messages can be represented in XML and encapsulated within  
248 HTTP messages. To attain interoperability among different implementations of CIM, both the XML  
249 representation and the HTTP encapsulation must be standardized. The XML representation is defined in  
250 [DSP0201](#), [DSP0203](#) and [DSP8044](#) define the DTD and XSD for that XML representation, for  
251 convenience. This document uses that XML representation to define the HTTP encapsulation.

252 The following criteria are applied to the representation of CIM-XML messages in XML using [DSP0201](#):

- 253 • Each CIM-XML message is completely described in XML; completeness is favored over  
254 conciseness.
- 255 • The set of CIM-XML messages provides enough functionality to enable implementations of CIM  
256 to communicate effectively for management purposes. This release of the mapping does not  
257 provide a *complete* set of messages. Rather, the goal is to define the mapping so that it admits  
258 straightforward extension (by the addition of further features) in future versions.
- 259 • The set of CIM-XML messages is classified into functional profiles to accommodate a range of  
260 implementations varying from complete support of all messages to support of a minimal subset.  
261 The number of functional profiles is kept as small as possible to encourage interoperability, and  
262 mechanisms provided by different CIM implementations can declare their level of support.

263 The following criteria are applied to the HTTP encapsulation of CIM-XML messages herein:

- 264 • In recognition of the large installed base of HTTP/1.0 systems, the encapsulation is designed to  
265 support both HTTP/1.0 and HTTP/1.1. However, support for HTTP/1.0 has been deprecated in  
266 version 1.4 of this document (see 7.1).
- 267 • The encapsulation does not introduce requirements that conflict with those stated in HTTP/1.0  
268 or HTTP/1.1.

- 269 • Use of the encapsulation should be straightforward over the current base HTTP infrastructures.  
270 Some features anticipate and exploit enhancements to this base, but no aspects of the  
271 encapsulation require such enhancements as mandatory.
- 272 • The encapsulation avoids the use of pure HTTP tunneling or URL munging (for example, the  
273 use of the "?" character) in favor of a mechanism that allows existing HTTP infrastructures to  
274 control content safely.
- 275 • The encapsulation exposes key CIM-XML message information in headers to allow efficient  
276 firewall/proxy handling. The information is limited to essentials so that it does not have a  
277 significant impact on the size of the header. All CIM-specific information in a header also  
278 appears within the CIM-XML message.
- 279 • There is a clear and unambiguous encapsulation of the CIM-XML message payload within the  
280 HTTP message. Conciseness of the encapsulation is of secondary importance.  
281



283

# CIM Operations over HTTP

## 284 1 Scope

285 The Common Information Model (CIM) (for details, see [DSP0004](#)) is an object-oriented information model  
286 defined by the Distributed Management Task Force (DMTF) that provides a conceptual framework for  
287 describing management data.

288 The Hypertext Transfer Protocol (HTTP) ([RFC1945](#), [RFC2616](#)) is an application-level protocol for  
289 distributed, collaborative, hypermedia information systems. This generic stateless protocol can be used  
290 for many tasks through extension of its request methods, error codes, and headers.

291 The Hypertext Transfer Protocol Secure (HTTPS) ([RFC2818](#)) is the usage of HTTP over secure sockets  
292 provided by TLS. It supports encryption of the messages exchanged, secure identification of servers, and  
293 secure authentication of clients.

294 NOTE: HTTPS should not be confused with Secure HTTP defined in RFC2660.

295 The [Extensible Markup Language \(XML\)](#) is a simplified subset of SGML that offers powerful and  
296 extensible data modeling capabilities. An *XML document* is a collection of data represented in XML. An  
297 *XML schema* is a grammar that describes the structure of an XML document.

298 This document defines a mapping of CIM-XML messages onto HTTP that allows implementations of CIM  
299 to interoperate in an open, standardized manner. It is based on [DSP0201](#) that defines the XML schema  
300 for CIM objects and messages.

## 301 2 Normative References

302 The following referenced documents are indispensable for applying the information in this document while  
303 developing an implementation of CIM. For dated references, only the edition cited applies. For undated  
304 references, the latest edition applies, including any amendments.

305 DMTF DSP0004, *Common Information Model (CIM) Infrastructure 2.7*,  
306 [http://www.dmtf.org/standards/published\\_documents/DSP0004\\_2.7.pdf](http://www.dmtf.org/standards/published_documents/DSP0004_2.7.pdf)

307 DMTF DSP0201, *Representation of CIM in XML 2.4*,  
308 [http://www.dmtf.org/standards/published\\_documents/DSP0201\\_2.4.pdf](http://www.dmtf.org/standards/published_documents/DSP0201_2.4.pdf)

309 DMTF DSP0212, *Filter Query Language 1.0*,  
310 [http://www.dmtf.org/standards/published\\_documents/DSP0212\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0212_1.0.pdf)

311 DMTF DSP0223, *Generic Operations 1.1*,  
312 [http://www.dmtf.org/standards/published\\_documents/DSP0223\\_1.1.pdf](http://www.dmtf.org/standards/published_documents/DSP0223_1.1.pdf)

313 DMTF DSP8016, *WBEM Operations Message Registry 1.1*,  
314 [http://www.dmtf.org/standards/published\\_documents/DSP8016\\_1.1.xml](http://www.dmtf.org/standards/published_documents/DSP8016_1.1.xml)

315 IETF RFC1766, *Tags for the Identification of Languages*, March 1995,  
316 <http://www.ietf.org/rfc/rfc1766.txt>

317 IETF RFC1945, *Hypertext Transfer Protocol – HTTP/1.0*, May 1996,  
318 <http://www.ietf.org/rfc/rfc1945.txt>

319 IETF RFC2246, *The TLS Protocol, Version 1.0*, January 1999,  
320 <http://www.ietf.org/rfc/rfc2246.txt>

- 321 IETF RFC2277, *IETF Policy on Character Sets and Languages*, January 1998,  
322 <http://www.ietf.org/rfc/rfc2277.txt>
- 323 IETF RFC2279, *UTF-8, a transformation format of Unicode and ISO 10646*, January 1998,  
324 <http://www.ietf.org/rfc/rfc2279.txt>
- 325 IETF RFC2376, *XML Media Types*, July 1998,  
326 <http://www.ietf.org/rfc/rfc2376.txt>
- 327 IETF RFC2396, *Uniform Resource Identifiers (URI): Generic Syntax*, August 1998,  
328 <http://www.ietf.org/rfc/rfc2396.txt>
- 329 IETF RFC2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999,  
330 <http://www.ietf.org/rfc/rfc2616.txt>
- 331 IETF RFC2617, *HTTP Authentication: Basic and Digest Access Authentication*, June 1999,  
332 <http://www.ietf.org/rfc/rfc2617.txt>
- 333 IETF RFC2774, *HTTP Extension Framework*, February 2000,  
334 <http://www.ietf.org/rfc/rfc2774.txt>
- 335 IETF RFC2818, *HTTP Over TLS*, May 2000,  
336 <http://www.ietf.org/rfc/rfc2818.txt>
- 337 IETF RFC4346, *The Transport Layer Security (TLS) Protocol, Version 1.1*, April 2006,  
338 <http://www.ietf.org/rfc/rfc4346.txt>
- 339 IETF RFC5246, *The Transport Layer Security (TLS) Protocol, Version 1.2*, August 2008,  
340 <http://www.ietf.org/rfc/rfc5246.txt>
- 341 NIST 800-57 Part 1, *Recommendation for Key Management: Part 1: General (Revision 3)*, July 2012,  
342 [http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57\\_part1\\_rev3\\_general.pdf](http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf)
- 343 NIST 800-131A, *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and  
344 Key Lengths*, January 2011,  
345 <http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf>
- 346 W3C Recommendation, *Extensible Markup Language (XML), Version 1.0*, August 2006,  
347 <http://www.w3.org/TR/REC-xml-names/>
- 348 W3C Recommendation, *Namespaces in XML*, January 1999,  
349 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
- 350 W3C, *XML Schema Part 1: Structures*, May 2001,  
351 <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>
- 352 W3C, *XSL Transformations (XSLT), Version 1.0*, November 1999,  
353 <http://www.w3.org/TR/xslt>
- 354 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*,  
355 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>

### 356 **3 Terms and Definitions**

357 In this document, some terms have a specific meaning beyond the normal English meaning. Those terms  
358 are defined in this clause.

359 The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"),  
360 "may", "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described

361 in [ISO/IEC Directives, Part 2](#), Annex H. The terms in parenthesis are alternatives for the preceding term,  
362 for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note  
363 that [ISO/IEC Directives, Part 2](#), Annex H specifies additional alternatives. Occurrences of such additional  
364 alternatives shall be interpreted in their normal English meaning.

365 The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as  
366 described in [ISO/IEC Directives, Part 2](#), Clause 5.

367 The terms "normative" and "informative" in this document are to be interpreted as described in [ISO/IEC](#)  
368 [Directives, Part 2](#), Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do  
369 not contain normative content. Notes and examples are always informative elements.

370 The terms defined in [DSP0004](#) and [DSP0201](#) apply to this document. The following additional terms are  
371 used in this document. Some additional more detailed terms are defined throughout the subclauses of  
372 this document.

### 373 3.1

#### 374 **CIM element**

375 one of the following components of the CIM metamodel used to define a schema: Class, instance,  
376 property, method, parameter, or qualifier

### 377 3.2

#### 378 **CIM object**

379 a namespace, class, instance, or qualifier that is accessible in a WBEM server

380

### 381 **CIM-XML protocol**

382 the WBEM protocol that uses the CIM operations over HTTP defined in this document and the  
383 representation of CIM in XML defined in [DSP0201](#)

### 384 3.3

#### 385 **WBEM client**

386 the client role in the CIM-XML protocol and in other WBEM protocols. See 6.1 for a complete definition.

### 387 3.4

#### 388 **WBEM listener**

389 the event listener role in the CIM-XML protocol and in other WBEM protocols. See 6.1 for a complete  
390 definition.

### 391 3.5

#### 392 **WBEM protocol**

393 a communications protocol between WBEM client, WBEM server and WBEM listener

### 394 3.6

#### 395 **WBEM server**

396 the server role in the CIM-XML protocol and in other WBEM protocols. See 6.1 for a complete definition.

### 397 3.7

#### 398 **XML element**

399 a component of XML that is defined using the ELEMENT construct in the DTD

## 400 **4 Abbreviated Terms and Document Conventions**

### 401 **4.1 Abbreviated Terms**

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

403

#### 404 **CIM**

405 Common Information Model

406

#### 407 **DTD**

408 Document Type Definition

409

#### 410 **HTTP**

411 Hypertext Transfer Protocol

412

#### 413 **XML**

414 Extensible Markup Language

### 415 **4.2 Document Conventions**

416 This document uses the same notational conventions and basic parsing constructs that are defined in  
417 [RFC2068](#).

418 Throughout this document, any deprecated element is indicated by one of the following labels:

- 419 • The “**DEPRECATION NOTE:**” label preceding a paragraph indicates that the paragraph  
420 explains a deprecated element.
- 421 • The “**DEPRECATED.**” label before a list item indicates that the information in that list item is  
422 deprecated.
- 423 • The “**(DEPRECATED)**” label after a heading applies to the entire clause for that heading.
- 424 • The “**(DEPRECATED)**” label at the end of a line in a code fragment or an example indicates that  
425 the particular line of the code fragment or example is deprecated.

## 426 **5 CIM-XML Message Syntax and Semantics**

427 This document defines all interactions among CIM products as CIM-XML messages. A CIM-XML  
428 message is a well-defined request or response data packet for exchanging information among CIM  
429 products. The two types of CIM-XML messages are as follows:

- 430 • *CIM-XML operation message.* This type of message is used between WBEM client and WBEM  
431 server to invoke an operation on the WBEM server.
- 432 • *CIM-XML export message.* This type of message is used between WBEM server and WBEM  
433 listener to communicate information (typically an event) to a WBEM listener.

434 This clause describes the syntax and semantics of CIM-XML messages independently of their  
435 encapsulation within a particular protocol such as HTTP. XML is used as the basis for this description,  
436 and in particular the CIM Representation in XML ([DSP0201](#)).

437 Note that "CIM message" (etc.) was used for the term "CIM-XML message" (etc.) before version 1.4 of  
438 this document.

## 439 5.1 Well-Formed, Valid, and Loosely Valid Documents

440 In this discussion, any reference to well-formed or valid XML documents has the standard meaning  
441 defined in [Extensible Markup Language \(XML\)](#).

442 XML document type definitions (DTDs) are restricted to be either well-formed or valid. However, this  
443 document also uses the term loosely valid to apply to XML that removes any attributes or elements in the  
444 XML document that do not appear in the [CIM XML DTD](#). The resulting document is valid with respect to  
445 the [CIM XML DTD](#) and is therefore loosely valid.

446 In effect, a loosely valid document is valid with respect to the [CIM XML DTD](#) apart from having additional  
447 attributes or elements not defined by that DTD. The concept is very similar to that of an open content  
448 model as defined by the working draft on [XML Schemas](#), expressed within the more limited scope of  
449 DTDs. One corollary of this definition is that any XML document that is valid with respect to the [CIM XML  
450 DTD](#) is also loosely valid.

451 The motivation for introducing the loosely valid class of XML documents is to relax the restrictions on a  
452 WBEM client, [WBEM server](#), or WBEM listener when parsing received XML documents defined within the  
453 scope of this mapping. Not all clients (including their respective WBEM servers or WBEM listeners)  
454 should be required to validate each received CIM-XML message response (or its respective CIM-XML  
455 message request) because such a requirement would place too heavy a processing burden on the  
456 validating entity at the expense of footprint and performance, most notably in communication between  
457 robust and conformant implementations of this mapping.

458 Instead, the following requirements are set forth in this document. In all cases, a WBEM client has a  
459 respective alternative WBEM server or WBEM listener, and a received CIM-XML message response has  
460 a respective alternative CIM-XML message request:

- 461 • A WBEM client may include a DOCTYPE element in a CIM-XML message request. If so, an  
462 external declaration should be used. In-lining of the complete DTD within a message is  
463 discouraged.
- 464 • A WBEM client may elect to validate a received CIM-XML message response.
- 465 • If a WBEM client elects not to validate a received CIM-XML message, then loose validation  
466 shall be enforced.

467 The behavior of a WBEM server or WBEM listener with respect to a received CIM-XML message request  
468 is covered in detail in 7.3.

## 469 5.2 Operational Semantics

470 The CIM Representation in XML ([DSP0201](#)) defines a child element under the root <CIM> XML element  
471 called <MESSAGE>, which contains one of the following XML child elements:

- 472 • CIM-XML operation message child elements
  - 473 – <SIMPLEREQ>
  - 474 – <SIMPLERSP>
  - 475 – <MULTIREQ>
  - 476 – <MULTIRSP>
- 477 • CIM-XML export message child elements
  - 478 – <SIMPLEXPREQ>

479           – <SIMPLEXPRES>

480           – <MULTIEXPREQ>

481           – <MULTIEXPRSP>

482 In the remainder of this document, the following terms denote an XML document that is loosely valid with  
483 respect to the CIM XML DTD:

484           • *Operation request message.* Contains under the root <CIM> node a <MESSAGE> child  
485 element that has a <MULTIREQ> or <SIMPLEREQ> child element under it.

486           • *Operation response message.* Contains under the root <CIM> node a <MESSAGE> child  
487 element that has a <MULTIRSP> or <SIMPLERSP> child element under it.

488           • *Export request message.* Contains under the root <CIM> node a <MESSAGE> child element  
489 that has a <MULTIEXPREQ> or <SIMPLEEXPREQ> child element under it.

490           • *Export response message.* Contains under the root <CIM> node a <MESSAGE> child element  
491 that has a <MULTIEXPRSP> or <SIMPLEEXPRSP> child element under it.

492 The phrase "CIM-XML message request" refers to either an operation request message or an export  
493 request message. The phrase "CIM-XML message response" refers to either an operation response  
494 message or an export response message.

495 A CIM-XML message request shall contain a non-empty value for the ID attribute of the <MESSAGE>  
496 element. The corresponding CIM-XML message response shall supply the same value for that attribute.  
497 Clients should employ a message ID scheme that minimizes the chance of receiving a stale CIM-XML  
498 message response.

499 Any CIM-XML message conforming to this document shall have a minimum value of "1.0" and a  
500 maximum value that is equal to the latest version of this document for the `PROTOCOLVERSION` attribute of  
501 the <MESSAGE> element.

502 An operation response message sent in response to an operation request message shall specify the  
503 same value for the `ID` attribute of the <MESSAGE> element that appears in the request message and  
504 contain one of the following:

505           – A <MULTIRSP> child element, if the operation request message contains a <MULTIREQ>  
506 child element.

507           – A <SIMPLERSP> child element, if the operation request message contains a  
508 <SIMPLEREQ> child element.

509 A *simple operation request* is an operation request message that contains a <SIMPLEREQ> child  
510 element. A simple operation response is an Operation Response Message that contains a  
511 <SIMPLERSP> child element.

512 A *multiple operation request* is an operation request message that contains a <MULTIREQ> child  
513 element. A multiple operation response is an operation response message that contains a <MULTIRSP>  
514 child element.

515 An export response message sent in response to an export request message shall specify the same  
516 value for the ID attribute of the <MESSAGE> element that appears in the export request message and  
517 shall contain one of the following:

518           – A <MULTIEXPRSP> child element if the export request message contained a  
519 <MULTIEXPREQ> child element, or

520           – A <SIMPLEEXPRSP> child element if the export request message contained a  
521 <SIMPLEEXPREQ> child element.

522 A simple export request is an export request message that contains a <SIMPLEXPREQ> child element.  
523 A simple export response is an export response message that contains a <SIMPLEXPRES> child  
524 element.

525 A multiple export request is an export request message that contains a <MULTIEXPREQ> child element.  
526 A multiple export response is an export response message that contains a <MULTIEXPRSP> child  
527 element.

## 528 **5.3 Operation Correlators**

### 529 **5.3.1 Overview**

530 WBEM servers may support maintaining a log to record certain aspects of operations requested by  
531 clients. The log data can provide a record of access, activity, configuration changes or audit related  
532 information. The purpose of audit related information is to identify what was done when servicing the  
533 operation, when it was done, and on behalf of which end user the operation was requested. In some  
534 environments, providing such audit information is a matter of regulatory compliance.

535 The credentials used for authentication with a WBEM server are not necessarily associated with the  
536 identity of an end user. For example, when the client application is a management server handling  
537 multiple end users, it is not uncommon to use the credentials of a system user (e.g. user "root" on Linux  
538 or UNIX systems) for authentication with the WBEM server. In such environments, a log on the WBEM  
539 server can only record the identity of the system user that was used for authentication, but not the identity  
540 of the end user on behalf of which the operation was requested.

541 Version 1.4 of this document introduced the concept of operation correlators which are named values that  
542 can be included by WBEM clients in operation request messages so that a WBEM server can add these  
543 correlators to any logs it maintains. To maintain symmetry, export request messages can also include  
544 operation correlators for use in any logs a WBEM listener may maintain.

545 The meaning of operation correlators is defined by the originator of the message and does not need to be  
546 understood by the receiver of the message; the receiver only stores the operation correlator along with  
547 any log entries about the message.

### 548 **5.3.2 Representation**

549 Operation correlators are represented in the CIM-XML protocol using the CORRELATOR element. Each  
550 occurrence of a CORRELATOR element represents one operation correlator. For details, see [DSP0201](#).

551 Zero or more operation correlators may be specified in simple operation request messages and in simple  
552 extrinsic request messages. Since the operations in a multiple operation may not have any semantic  
553 relationship within each other, the operation correlators are specified only at the level of simple operations  
554 within the multiple operation; operation correlators cannot be specified at the level of multiple operations.

555 This document defines no requirements on the number, content or meaning of operation correlators.

### 556 **5.3.3 Implementation Requirements and Compatibility for Operation Messages**

557 Supporting operation correlators for WBEM clients is optional. If a WBEM client implements support for  
558 operation correlators, it may include zero or more operation correlators in a simple operation request  
559 message. The number, content and meaning of operation correlators may be different in each operation.

560 Supporting operation correlators for WBEM servers for its operation messages is optional. If a WBEM  
561 server implements support for operation correlators for its operation messages, it shall store the operation  
562 correlators specified in a simple operation request message along with any log information about the  
563 operation. If the operation itself is not logged on the server, the correlator also does not need to be

564 logged. In order to avoid vulnerabilities by specification of excessive amounts of operation correlators,  
565 WBEM servers may implement limits on operation correlators.

566 Since participants in the protocol defined by this document are required to ignore any unknown XML  
567 elements in messages they receive, introducing support for operation correlators in WBEM clients is  
568 compatible for WBEM servers that do not support them.

### 569 **5.3.4 Implementation Requirements and Compatibility for Export Messages**

570 Supporting operation correlators for WBEM servers for its export messages is optional. If a WBEM  
571 server implements support for operation correlators for its export messages, it may include zero or more  
572 operation correlators in a simple export request message. The number, content and meaning of operation  
573 correlators may be different in each export message.

574 Supporting operation correlators for WBEM listeners is optional. If a WBEM listener implements support  
575 for operation correlators, it shall store the operation correlators specified in a simple export request  
576 message along with any log information about the export message. If the export message itself is not  
577 logged on the listener, the correlator also does not need to be logged. In order to avoid vulnerabilities by  
578 specification of excessive amounts of operation correlators, WBEM listeners may implement limits on  
579 operation correlators.

580 Since participants in the protocol defined by this document are required to ignore any unknown XML  
581 elements in messages they receive, introducing support for operation correlators in WBEM servers for its  
582 export messages is compatible for WBEM listeners that do not support them.

## 583 **5.4 CIM Operation Syntax and Semantics**

584 This clause describes method invocations, intrinsic methods, and namespace manipulation.

### 585 **5.4.1 Method Invocations**

586 All CIM-XML operation requests defined for this CIM-to-HTTP mapping are defined as invocations of one  
587 or more methods. A method can be either:

- 588 • An *intrinsic* method, which is defined for the purposes of modeling a CIM operation.
- 589 • An *extrinsic* method, which is defined on a CIM class in a schema.

590 In addition, intrinsic methods are made against a CIM namespace. Extrinsic methods are invoked on a  
591 CIM class (if static) or instance otherwise. Intrinsic methods are defined in 5.4.2.

592 An extrinsic method call is represented in XML by the <METHODCALL> element, and the response to  
593 that call is represented by the <METHODRESPONSE> element.

594 An intrinsic method call is represented in XML by the <IMETHODCALL> element, and the response to  
595 that call is represented by the <IMETHODRESPONSE> element. An input parameter has an IN qualifier  
596 (with a value of `true`) in the method definition, and an output parameter has an OUT qualifier (with a  
597 value of `true`). A parameter can be both an input and an output parameter.

598 The <METHODCALL> or <IMETHODCALL> element names the method to be invoked and supplies any  
599 input parameters to the method call. Note the following rules about parameters:

- 600 • Each input parameter shall be named using the name assigned in the method definition.
- 601 • Input parameters may be supplied in any order.
- 602 • Each input parameter of the method, and no others, shall be present in the call, unless it is  
603 optional.

604 The <METHODRESPONSE> or <IMETHODRESPONSE> element defines either an <ERROR> or an  
605 optional return value and output parameters if it is decorated with the OUT qualifier in the method  
606 definition. In the latter case, the following rules about parameters apply:

- 607 • Each output parameter shall be named using the name assigned in the method definition.
- 608 • Output parameters may be supplied in any order.
- 609 • Each output parameter of the method, and no others, shall be present in the response, unless it  
610 is optional.
- 611 • The method invocation process can be thought of as the binding of the input parameter values  
612 specified as child elements of the <METHODCALL> or <IMETHODCALL> element to the input  
613 parameters of the method. This binding is followed by an attempt to execute the method using  
614 the bound input parameters with one of the following results:
  - 615 – If the attempt to call the method is successful, the return value and output parameters are  
616 bound to the child elements of the <METHODRESPONSE> or <IMETHODRESPONSE>  
617 element.
  - 618 – If the attempt to call the method is unsuccessful, an error code and optional human-  
619 readable description of that code is bound to the <METHODRESPONSE> or  
620 <IMETHODRESPONSE> element.

#### 621 5.4.1.1 Simple Operations

622 A simple operation invokes a single method. A simple operation request is represented by a  
623 <SIMPLEREQ> element, and a simple operation response is represented by a <SIMPLERSP> element.

624 If the method is [intrinsic](#), then the <SIMPLEREQ> element shall contain an <IMETHODCALL> element,  
625 which in turn contains a <LOCALNAMESPACEPATH> child element identifying the local CIM namespace  
626 against which the method is to execute. If the method is [extrinsic](#), then the <SIMPLEREQ> element shall  
627 contain a <METHODCALL> element that in turn contains one of the following child elements:

- 628 • A <LOCALCLASSPATH> child element identifying the CIM class on which the method is to be  
629 invoked if the method is static.
- 630 • A <LOCALINSTANCEPATH> child element identifying the CIM instance on which the method is  
631 otherwise to be invoked.

#### 632 5.4.1.2 Multiple Operations

633 A multiple operation requires the invocation of more than one method. A multiple operation request is  
634 represented by a <MULTIREQ> element, and a multiple operation response is represented by a  
635 <MULTIRSP> element.

636 A <MULTIREQ> (or its respective <MULTIRSP>) element is a sequence of two or more <SIMPLEREQ>  
637 (or their respective <SIMPLERSP>) elements.

638 A <MULTIRSP> element shall contain a <SIMPLERSP> element for every <SIMPLEREQ> element in the  
639 corresponding multiple operation response. These <SIMPLERSP> elements shall be in the same order  
640 as their <SIMPLEREQ> counterparts so that the first <SIMPLERSP> in the response corresponds to the  
641 first <SIMPLEREQ> in the request, and so forth.

642 Multiple operations conveniently allow multiple method invocations to be batched into a single HTTP  
643 message. Batching reduces the number of roundtrips between a [WBEM client](#) and a WBEM server and  
644 allows the WBEM server to make internal optimizations if it chooses. Note that multiple operations do not  
645 confer any transactional capabilities in processing the request. For example, the WBEM server does not  
646 have to guarantee that the constituent method calls either all fail or succeed, only that the entity make a  
647 "best effort" to process the operation. However, servers shall finish processing each operation in a

648 batched operation before executing the next one. Clients shall recognize that the order of operations  
649 within a batched operation is significant.

650 Not all WBEM servers support multiple operations; the way they declare support for this feature is defined  
651 in 7.5.

### 652 5.4.1.3 Status Codes

653 This clause defines the status codes and detailed error information that a conforming WBEM server  
654 application can return. The value of an <ERROR> child element within a <METHODRESPONSE> or  
655 <IMETHODRESPONSE> element includes the following parts:

- 656 • a mandatory status code
- 657 • an optional human-readable description of the status code
- 658 • zero or more CIM\_Error instances

659 Table 1 defines the status codes that a conforming WBEM server application can return as the value of  
660 the CODE attribute of an <ERROR> child element. In addition to a status code, a conforming WBEM  
661 server may return zero or more <INSTANCE> child elements as part of an <ERROR> element. Each  
662 <INSTANCE> child element shall be an instance of CIM\_Error. For each instance of CIM\_Error, the value  
663 of CIMStatusCode shall comply with the definition of expected error codes for the CIM-XML operation  
664 request. A WBEM client may ignore any <INSTANCE> child elements.

665 The symbolic names defined in Table 1 do not appear on the wire. They are used here solely for  
666 convenient reference to an error in other parts of this document.

667 Not all methods are expected to return all the status codes listed in Table 1. For [intrinsic methods](#), the  
668 relevant clause on each method in this document defines the error codes expected to be returned. For  
669 extrinsic methods, 5.4.5 specifies which of the codes in Table 1 can be used.

Table 1 – Status Codes Returned by an &lt;Error&gt; Child element

Symbolic Name	Code	Definition
CIM_ERR_FAILED	1	A general error occurred that is not covered by a more specific error code.
CIM_ERR_ACCESS_DENIED	2	Access to a CIM resource is not available to the client.
CIM_ERR_INVALID_NAMESPACE	3	The target namespace does not exist.
CIM_ERR_INVALID_PARAMETER	4	One or more parameter values passed to the method are not valid.
CIM_ERR_INVALID_CLASS	5	The specified class does not exist.
CIM_ERR_NOT_FOUND	6	The requested object cannot be found. The operation can be unsupported on behalf of the WBEM server in general or on behalf of an implementation of a management profile.
CIM_ERR_NOT_SUPPORTED	7	The requested operation is not supported on behalf of the WBEM server, or on behalf of a provided class. If the operation is supported for a provided class but is not supported for particular instances of that class, then CIM_ERR_FAILED shall be used.
CIM_ERR_CLASS_HAS_CHILDREN	8	The operation cannot be invoked on this class because it has subclasses.
CIM_ERR_CLASS_HAS_INSTANCES	9	The operation cannot be invoked on this class because one or more instances of this class exist.
CIM_ERR_INVALID_SUPERCLASS	10	The operation cannot be invoked because the specified superclass does not exist.
CIM_ERR_ALREADY_EXISTS	11	The operation cannot be invoked because an object already exists.
CIM_ERR_NO_SUCH_PROPERTY	12	The specified property does not exist.
CIM_ERR_TYPE_MISMATCH	13	The value supplied is not compatible with the type.
CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED	14	The query language is not recognized or supported.
CIM_ERR_INVALID_QUERY	15	The query is not valid for the specified query language.
CIM_ERR_METHOD_NOT_AVAILABLE	16	The extrinsic method cannot be invoked.
CIM_ERR_METHOD_NOT_FOUND	17	The specified extrinsic method does not exist.

Symbolic Name	Code	Definition
CIM_ERR_NAMESPACE_NOT_EMPTY	20	The specified namespace is not empty.
CIM_ERR_INVALID_ENUMERATION_CONTEXT	21	The enumeration identified by the specified context cannot be found, is in a closed state, does not exist, or is otherwise invalid.
CIM_ERR_INVALID_OPERATION_TIMEOUT	22	The specified operation timeout is not supported by the WBEM server.
CIM_ERR_PULL_HAS_BEEN_ABANDONED	23	The Pull operation has been abandoned due to execution of a concurrent CloseEnumeration operation on the same enumeration.
CIM_ERR_PULL_CANNOT_BE_ABANDONED	24	The attempt to abandon a concurrent Pull operation on the same enumeration failed. The concurrent Pull operation proceeds normally.
CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED	25	Using a filter query in pulled enumerations is not supported by the WBEM server.
CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED	26	The WBEM server does not support continuation on error.
CIM_ERR_SERVER_LIMITS_EXCEEDED	27	The WBEM server has failed the operation based upon exceeding server limits.
CIM_ERR_SERVER_IS_SHUTTING_DOWN	28	The WBEM server is shutting down and cannot process the operation.

#### 671 5.4.2 Intrinsic Methods

672 This clause describes the [Intrinsic](#) methods defined outside the schema for CIM operations. These  
673 methods can only be called on a CIM namespace, rather than on a CIM class or instance.

674 The notation used in the following subclauses to define the signatures of the intrinsic methods is a  
675 pseudo-MOF notation that extends the standard MOF BNF ([DSP0004](#)) for describing CIM methods with  
676 several pseudo-parameter types enclosed within angle brackets (< and >).

677 This notation decorates the parameters with pseudo-qualifiers (IN, OUT, OPTIONAL, and NULL) to define  
678 their invocation semantics. These qualifiers are for description purposes only within the scope of this  
679 document; in particular, a [WBEM client](#) shall not specify them in intrinsic method invocations.

680 This notation uses the IN qualifier to denote that the parameter is an input parameter.

681 This notation uses the OUT qualifier to denote that the parameter is an output parameter.

682 A WBEM client may omit an optional parameter by not specifying an <IPARAMVALUE> element for that  
683 parameter if the required value is the specified default. It shall not omit a parameter that is not marked as  
684 optional. A WBEM server may omit support for an optional parameter. Any attempt to call a method with  
685 an optional parameter that is not supported shall return either CIM\_ERR\_NOT\_SUPPORTED or  
686 CIM\_ERR\_INVALID\_PARAMETER.

687 This notation uses the NULL qualifier for parameters whose values can be specified as NULL in a method  
688 call. A NULL (unassigned) value for a parameter is specified by an <IPARAMVALUE> or

689 <PARAMVALUE> element with no child element. For parameters without the NULL qualifier, the WBEM  
 690 client shall specify a value by including a suitable child element for the <IPARAMVALUE> or  
 691 <PARAMVALUE> element.

692 All parameters shall be uniquely named and shall correspond to a valid parameter name for that method  
 693 as described by this document. The order of the parameters is not significant.

694 The non-NULL values of intrinsic method parameters or return values modeled as standard CIM types  
 695 (such as string and Boolean or arrays thereof) are represented as follows:

- 696 • Simple values use the <VALUE> child element within an <IPARAMETER> element for method  
 697 parameters or within an <IRETURNVALUE> element for method return values.
- 698 • Array values use the <VALUE.ARRAY> child element within an <IPARAMETER> element for  
 699 method parameters or within an <IRETURNVALUE> element for method return values.

700 Table 2 shows how each pseudo-type used by the intrinsic methods shall be mapped to an XML element  
 701 described in [DSP0201](#) in the context of both a parameter value (child element of <IPARAMVALUE>) and  
 702 a return value (child element of <IRETURNVALUE>).

703 **Table 2 – Mapping of Intrinsic Method Pseudo-Types to XML Elements**

Type	XML Element
<object>	(VALUE.OBJECT VALUE.OBJECTWITHLOCALPATH VALUE.OBJECTWITHPATH)
<class>	CLASS
<instance>	INSTANCE
<className>	CLASSNAME
<namedInstance>	VALUE.NAMEDINSTANCE
<instanceName>	INSTANCENAME
<instancePath>	INSTANCEPATH
<objectWithPath>	VALUE.OBJECTWITHPATH
<instanceWithPath>	VALUE.INSTANCEWITHPATH
<objectName>	(CLASSNAME INSTANCENAME)
<objectPath>	OBJECTPATH
<propertyValue>	(VALUE VALUE.ARRAY VALUE.REFERENCE)
<qualifierDecl>	QUALIFIER.DECLARATION

704 **5.4.2.1 GetClass**

705 The GetClass operation returns a single CIM class from the target namespace:

```

706 <class> GetClass (
707     [IN] <className> ClassName,
708     [IN,OPTIONAL] boolean LocalOnly = true,
709     [IN,OPTIONAL] boolean IncludeQualifiers = true,
710     [IN,OPTIONAL] boolean IncludeClassOrigin = false,
711     [IN,OPTIONAL,NULL] string PropertyList [] = NULL
712 )
    
```

713 The `ClassName` input parameter defines the name of the class to be retrieved.

714 If the `LocalOnly` input parameter is `true`, any CIM elements (properties, methods, and qualifiers),  
 715 except those added or overridden in the class as specified in the `classname` input parameter, shall not be  
 716 included in the returned class. If it is `false`, no additional filtering is defined.

717 If the `IncludeQualifiers` input parameter is `true`, all qualifiers for that class (including qualifiers on  
 718 the class and on any returned properties, methods, or method parameters) shall be included as  
 719 `<QUALIFIER>` XML elements in the response. If it is `false`, no `<QUALIFIER>` XML elements are present  
 720 in the returned class.

721 If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on  
 722 all appropriate elements in the returned class. If it is `false`, no `CLASSORIGIN` attributes are present in  
 723 the returned class.

724 If the `PropertyList` input parameter is not `NULL`, the members of the array define one or more property  
 725 names. The returned class shall not include any properties missing from this list. Note that if `LocalOnly`  
 726 is specified as `true`, it acts as an additional filter on the set of properties returned. For example, if  
 727 property `A` is included in `PropertyList` but `LocalOnly` is set to `true` and `A` is not local to the  
 728 requested class, it is not included in the response. If the `PropertyList` input parameter is an empty  
 729 array, no properties are included in the response. If the `PropertyList` input parameter is `NULL`, no  
 730 additional filtering is defined.

731 If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise  
 732 process the request normally. If `PropertyList` contains property names that are invalid for the target  
 733 class, the WBEM server shall ignore them but otherwise process the request normally.

734 If `GetClass` is successful, the return value is a single CIM class that shall include all CIM elements  
 735 (properties, methods, and qualifiers) defined in or inherited by that class, reduced by any elements  
 736 excluded as a result of using the `LocalOnly` or `PropertyList` filters.

737 If `GetClass` is unsuccessful, this method shall return one of the following status codes, where the error  
 738 returned is the first applicable error in the list, starting with the first element and working down. Any  
 739 additional method-specific interpretation of the error is enclosed in parentheses:

- 740 • `CIM_ERR_ACCESS_DENIED`
- 741 • `CIM_ERR_INVALID_NAMESPACE`
- 742 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized or otherwise  
 743 incorrect parameters)
- 744 • `CIM_ERR_NOT_FOUND` (The request CIM class does not exist in the specified namespace.)
- 745 • `CIM_ERR_FAILED` (Some other unspecified error occurred.)

#### 746 5.4.2.2 `GetInstance`

747 The `GetInstance` operation returns a single CIM instance from the target namespace:

```
748 <instance> GetInstance (
749     [IN] <instanceName> InstanceName,
750     [IN,OPTIONAL] boolean LocalOnly = true,           (DEPRECATED)
751     [IN,OPTIONAL] boolean IncludeQualifiers = false, (DEPRECATED)
752     [IN,OPTIONAL] boolean IncludeClassOrigin = false,
753     [IN,OPTIONAL,NULL] string PropertyList [] = NULL
754 )
```

755 The `InstanceName` input parameter defines the name of the instance to be retrieved.

756 **DEPRECATION NOTE:** With version 1.2 of this document, the `LocalOnly` parameter is DEPRECATED.  
757 `LocalOnly` filtering, as defined in 1.1, will not be supported in the next major revision of this document.  
758 In version 1.1 of this document, the definition of the `LocalOnly` parameter was incorrectly modified. This  
759 change introduced a number of interoperability and backward compatibility problems for WBEM clients  
760 using the `LocalOnly` parameter to filter the set of properties returned. The DMTF strongly recommends  
761 that WBEM clients set `LocalOnly` to `false` and do not use this parameter to filter the set of properties  
762 returned. To minimize the impact of this recommendation on WBEM clients, a WBEM server may choose  
763 to treat the value of the `LocalOnly` parameter as `false` for all requests. A WBEM server shall  
764 consistently support a single interpretation of the `LocalOnly` parameter. Refer to ANNEX B for additional  
765 details.

766 **DEPRECATION NOTE:** The use of the `IncludeQualifiers` parameter is DEPRECATED and it may  
767 be removed in a future version of this document. The `IncludeQualifiers` parameter definition is  
768 ambiguous and when it is set to `true`, WBEM clients cannot be assured that any qualifiers will be  
769 returned. A WBEM client should always set `IncludeQualifiers` to `false`. To minimize the impact of  
770 this recommendation on WBEM clients, a WBEM server may choose to treat the value of the  
771 `IncludeQualifiers` parameter as `false` for all requests. The preferred behavior is to use the class  
772 operations to receive qualifier information and not depend on any qualifiers existing in this response. If  
773 the `IncludeQualifiers` input parameter is `true`, all qualifiers for that instance (including qualifiers on  
774 the instance and on any returned properties) shall be included as `<QUALIFIER>` XML elements in the  
775 response. If it is `false`, no `<QUALIFIER>` XML elements are present.

776 If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on  
777 all appropriate elements in the returned instance. If it is `false`, no `CLASSORIGIN` attributes are present.

778 If the `PropertyList` input parameter is not `NULL`, the members of the array define one or more property  
779 names. The returned instance shall not include any properties missing from this list. Note that if  
780 `LocalOnly` is `true`, this acts as an additional filter on the set of properties returned. For example, if  
781 property `A` is included in `PropertyList` but `LocalOnly` is set to `true` and `A` is not local to the  
782 requested instance, it is not included in the response. If the `PropertyList` input parameter is an empty  
783 array, no properties are included in the response. If the `PropertyList` input parameter is `NULL`, no  
784 additional filtering is defined.

785 If `PropertyList` contains duplicate property names, the WBEM server shall ignore the duplicates but  
786 otherwise process the request normally. If `PropertyList` contains property names that are invalid for  
787 the target instance, the WBEM server shall ignore them but otherwise process the request normally.

788 Properties with the `NULL` value may be omitted from the response, even if the WBEM client has not  
789 requested the exclusion of the property through the `LocalOnly` or `PropertyList` filters. The WBEM  
790 client shall interpret such omitted properties as `NULL`. Note that the WBEM client cannot make any  
791 assumptions about properties omitted as a result of using `LocalOnly` or `PropertyList` filters.

792 If `GetInstance` is successful, the return value is a single CIM instance with all properties defined in and  
793 inherited by its class reduced by any properties excluded as a result of using the `LocalOnly` or  
794 `PropertyList` filters and further reduced by any `NULL` valued properties omitted from the response.

795 If `GetInstance` is unsuccessful, the method shall return one of the following status codes where the error  
796 returned is the first applicable error in the list, starting with the first element and working down. Any  
797 additional method-specific interpretation of the error is enclosed in parentheses:

- 798 • `CIM_ERR_ACCESS_DENIED`
- 799 • `CIM_ERR_INVALID_NAMESPACE`
- 800 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
801 incorrect parameters)

- 802 • CIM\_ERR\_INVALID\_CLASS (The CIM class does not exist in the specified namespace.)
- 803 • CIM\_ERR\_NOT\_FOUND (The CIM class does exist, but the requested CIM instance does not  
804 exist in the specified namespace.)
- 805 • CIM\_ERR\_FAILED (some other unspecified error occurred)

#### 806 5.4.2.3 DeleteClass

807 The DeleteClass operation deletes a single CIM class from the target namespace:

```
808     void DeleteClass (
809         [IN] <className> ClassName
810     )
```

811 The `ClassName` input parameter defines the name of the class to be deleted.

812 If DeleteClass is successful, the WBEM server removes the specified class, including any subclasses and  
813 any instances. The operation shall fail if any one of these objects cannot be deleted.

814 If DeleteClass is unsuccessful, this method shall return one of the following status codes, where the error  
815 returned is the first applicable error in the list, starting with the first element and working down. Any  
816 additional method-specific interpretation of the error is enclosed in parentheses:

- 817 • CIM\_ERR\_ACCESS\_DENIED
- 818 • CIM\_ERR\_NOT\_SUPPORTED
- 819 • CIM\_ERR\_INVALID\_NAMESPACE
- 820 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
821 incorrect parameters)
- 822 • CIM\_ERR\_NOT\_FOUND (The CIM class to be deleted does not exist.)
- 823 • CIM\_ERR\_CLASS\_HAS\_CHILDREN (The CIM class has one or more subclasses that cannot  
824 be deleted.)
- 825 • CIM\_ERR\_CLASS\_HAS\_INSTANCES (The CIM class has one or more instances that cannot  
826 be deleted.)
- 827 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 828 5.4.2.4 DeleteInstance

829 The DeleteInstance operation deletes a single CIM instance from the target namespace.

```
830     void DeleteInstance (
831         [IN] <instanceName> InstanceName
832     )
```

833 The `InstanceName` input parameter defines the name (model path) of the instance to be deleted.

834 Deleting the instance may or may not cause the automatic deletion of additional instances. For example,  
835 the deletion of an instance may cause the automatic deletion of all associations that reference that  
836 instance. Or the deletion of an instance may cause the automatic deletion of instances (and their  
837 associations) that have a Min(1) relationship to that instance.

838 If DeleteInstance is successful, the WBEM server removes the specified instance.

839 If DeleteInstance is unsuccessful, this method shall return one of the following status codes, where the  
 840 error returned is the first applicable error in the list, starting with the first element and working down. Any  
 841 additional method-specific interpretation of the error is enclosed in parentheses.

- 842 • CIM\_ERR\_ACCESS\_DENIED
- 843 • CIM\_ERR\_NOT\_SUPPORTED (by the WBEM server for this operation)
- 844 • CIM\_ERR\_INVALID\_NAMESPACE
- 845 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
 846 incorrect parameters)
- 847 • CIM\_ERR\_INVALID\_CLASS (The CIM class does not exist in the specified namespace.)
- 848 • CIM\_ERR\_NOT\_SUPPORTED (This operation is not supported for the class of the specified  
 849 instance, if provided.)
- 850 • CIM\_ERR\_NOT\_FOUND (The CIM class does exist, but the requested CIM instance does not  
 851 exist in the specified namespace.)
- 852 • CIM\_ERR\_FAILED (This operation is not supported for the specified instance, or some other  
 853 unspecified error occurred.)

#### 854 5.4.2.5 CreateClass

855 The CreateClass operation creates a single CIM class in the target namespace. The class shall not  
 856 already exist:

```
857     void CreateClass (
858         [IN] <class> NewClass
859     )
```

860 The `NewClass` input parameter defines the new class. The proposed definition shall be a correct class  
 861 definition according to [DSP0004](#).

862 In processing the creation of the new class, the WBEM server shall conform to the following rules:

- 863 • The server shall ignore any CLASSORIGIN and PROPAGATED XML attributes in the new  
 864 class.
- 865 • If the new class has no superclass, the `NewClass` parameter defines a new superclass. The  
 866 server shall ensure that all properties and methods of the new class have a CLASSORIGIN  
 867 attribute whose value is the name of the new class.
- 868 • If the new class has a superclass, the `NewClass` parameter defines a new subclass of that  
 869 superclass. The superclass shall exist. The server shall ensure that the following conditions are met:  
 870
  - 871 – Any properties, methods, or qualifiers in the subclass not defined in the superclass are  
 872 created as new elements of the subclass. In particular, the server shall set the  
 873 CLASSORIGIN XML attribute on the new properties and methods to the name of the  
 874 subclass and ensure that all others preserve their CLASSORIGIN attribute value from that  
 875 defined in the superclass.
  - 876 – If a property is defined in the superclass and in the subclass, the value assigned to that  
 877 property in the subclass (including NULL) becomes the default value of the property for the  
 878 subclass.
  - 879 – If a property or method of the superclass is not specified in the subclass, then it is inherited  
 880 without modification by the subclass.

- 881           – Any qualifiers defined in the superclass with a TOSUBCLASS attribute value of `true` shall  
 882           appear in the resulting subclass. Qualifiers in the superclass with a TOSUBCLASS  
 883           attribute value of `false` shall not be propagated to the subclass.
- 884           – Any qualifier propagated from the superclass cannot be modified in the subclass if the  
 885           OVERRIDEABLE attribute of that qualifier is set to `false` in the superclass. It is a client  
 886           error to specify such a qualifier in the new class with a definition different than that in the  
 887           superclass (where definition encompasses the name, type, and flavor attribute settings of  
 888           the <QUALIFIER> XML element and the value of the qualifier).

889   If CreateClass is successful, the WBEM server creates the specified class.

890   If CreateClass is unsuccessful, this method shall return one of the following status codes, where the error  
 891   returned is the first applicable error in the list, starting with the first element and working down. Any  
 892   additional method-specific interpretation of the error is enclosed in parentheses.

- 893           • CIM\_ERR\_ACCESS\_DENIED
- 894           • CIM\_ERR\_NOT\_SUPPORTED
- 895           • CIM\_ERR\_INVALID\_NAMESPACE
- 896           • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
 897           incorrect parameters)
- 898           • CIM\_ERR\_ALREADY\_EXISTS (The CIM class already exists.)
- 899           • CIM\_ERR\_INVALID\_SUPERCLASS (The putative CIM class declares a non-existent  
 900           superclass.)
- 901           • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 902   5.4.2.6   CreateInstance

903   The CreateInstance operation creates a single CIM Instance in the target namespace. The instance shall  
 904   not already exist:

```
905       <instanceName> CreateInstance (
906           [IN] <instance> NewInstance
907       )
```

908   **DEPRECATION NOTE:** The use of qualifiers on instances is DEPRECATED and may be removed in a  
 909   future version of this document. A WBEM client cannot rely on any qualifiers included in the  
 910   NewInstance to have any impact on the operation. It is recommended that the WBEM server ignore any  
 911   qualifiers included in the instance. The NewInstance input parameter defines the new instance. The  
 912   proposed definition shall be a correct instance definition for the underlying CIM class according to  
 913   [DSP0004](#).

914   In creating the new instance, the WBEM server shall conform to the following rules and ensure that they  
 915   are applied:

- 916           • The server shall ignore any CLASSORIGIN and PROPAGATED XML attributes in the  
 917           NewInstance.
- 918           • **DEPRECATED.** Any qualifiers in the instance not defined in the class are created as new  
 919           elements of the instance.
- 920           • All properties of the instance preserve their CLASSORIGIN attribute value from that defined in  
 921           the class.
- 922           • The designated initial value for any property in the CIM instance to be created shall be the  
 923           property value (including NULL) specified in the NewInstance parameter, or if the property is

- 924 not specified in the `NewInstance` parameter, the default value (including NULL) defined in the  
 925 property declaration, or if the property does not define a default value, there is no designated  
 926 initial value for the property.
- 927 If there is a designated initial value for a property, the server shall either initialize the property to  
 928 that value, or reject the request. If there is no designated initial value for a property, the server  
 929 may initialize the property to any value (including NULL). Further considerations for accepting or  
 930 rejecting creation requests based on the properties requested to be initialized are out of scope  
 931 for this document; CIM model definitions are expected to cover that.
- 932 • If the `NewInstance` parameter specifies properties that are not exposed by the class specified  
 933 in the `NewInstance` parameter, the server shall reject the request.
  - 934 • **DEPRECATION NOTE:** Use of the `TOINSTANCE` attribute is DEPRECATED. Servers may  
 935 choose to ignore `TOINSTANCE`. Servers that do not ignore `TOINSTANCE` shall interpret it so  
 936 that any qualifiers defined in the class with a `TOINSTANCE` attribute value of `true` appear in  
 937 the instance. Qualifiers in the class with a value of `false` shall not be propagated to the  
 938 instance.
  - 939 • **DEPRECATED.** Any Qualifier propagated from the class cannot be modified in the instance if  
 940 the `OVERRIDABLE` attribute of that qualifier is set to `false` in the class. It is a client error to  
 941 specify such a qualifier in the `NewInstance` with a definition different than that in the class  
 942 (where definition encompasses the name, type, and flavor attribute settings of the  
 943 `<QUALIFIER>` XML element and the value of the qualifier).

944 If `CreateInstance` is successful, the new CIM instance has been created as described in this subclause,  
 945 and the return value defines the object path of the new CIM instance relative to the target namespace  
 946 created by the WBEM server (that is, the model path as defined by [DSP0004](#)). It is returned if one or  
 947 more of the new keys of the instance are dynamically allocated during creation rather than specified in the  
 948 request.

949 If `CreateInstance` is unsuccessful, this method shall return one of the following status codes, where the  
 950 error returned is the first applicable error in the list, starting with the first element and working down. Any  
 951 additional method-specific interpretation of the error is enclosed in parentheses.

- 952 • `CIM_ERR_ACCESS_DENIED`
- 953 • `CIM_ERR_NOT_SUPPORTED` (by the WBEM server for this operation)
- 954 • `CIM_ERR_INVALID_NAMESPACE`
- 955 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
 956 incorrect parameters)
- 957 • `CIM_ERR_INVALID_CLASS` (The CIM class for the new instance does not exist.)
- 958 • `CIM_ERR_NOT_SUPPORTED` (This operation is not supported for the class of the specified  
 959 instance, if provided.)
- 960 • `CIM_ERR_ALREADY_EXISTS` (The CIM instance already exists.)
- 961 • `CIM_ERR_FAILED` (This operation is not supported for the specified instance or some other  
 962 unspecified error occurred.)

#### 963 5.4.2.7 **ModifyClass**

964 The `ModifyClass` operation modifies an existing CIM class in the target namespace. The class shall  
 965 already exist:

```
966     void ModifyClass (
967         [IN] <class> ModifiedClass
```

968 )

969 The `ModifiedClass` input parameter defines the set of changes to be made to the current class  
970 definition, which shall be correct amendments to the CIM class as defined by [DSP0004](#).

971 In modifying the class, the WBEM server shall conform to the following rules:

- 972 • The WBEM server shall ignore any `CLASSORIGIN` and `PROPAGATED` XML attributes in the  
973 `ModifiedClass`.
- 974 • If the modified class has no superclass, the `ModifiedClass` parameter defines modifications to a  
975 superclass. The server shall ensure that the following conditions are met:
  - 976 – All properties and methods of the modified class have a `CLASSORIGIN` attribute whose  
977 value is the name of this class.
  - 978 – Any properties, methods, or qualifiers in the existing class definition that do not appear in  
979 the `ModifiedClass` parameter are removed from the resulting modified class.
- 980 • If the modified class has a superclass, the `ModifiedClass` parameter defines modifications to  
981 a subclass of that superclass. The superclass shall exist, and the client shall not change the  
982 name of the superclass in the modified subclass. The server shall ensure that the following  
983 conditions are met:
  - 984 – Any properties, methods, or qualifiers in the subclass not defined in the superclass are  
985 created as elements of the subclass. In particular, the server shall set the `CLASSORIGIN`  
986 attribute on the new properties and methods to the name of the subclass and shall ensure  
987 that all other others preserve their `CLASSORIGIN` attribute value from that defined in the  
988 superclass.
  - 989 – Any property, method, or qualifier previously defined in the subclass but not defined in the  
990 superclass, and which is not present in the `ModifiedClass` parameter, is removed from  
991 the subclass.
  - 992 – If a property is specified in the `ModifiedClass` parameter, the value assigned to that  
993 property (including `NULL`) becomes the default value of the property for the subclass.
  - 994 – If a property or method of the superclass is not specified in the subclass, then the subclass  
995 inherits it without modification. Any previous changes to such an element in the subclass  
996 are lost.
  - 997 – If a qualifier in the superclass is not specified in the subclass and the qualifier is defined in  
998 the superclass with a `TOSUBCLASS` attribute value of `true`, then the qualifier shall still be  
999 present in the resulting modified subclass. A propagated qualifier cannot be removed from  
1000 a subclass.
  - 1001 – Any qualifier propagated from the superclass cannot be modified in the subclass if the  
1002 `OVERRIDEABLE` attribute of that qualifier is set to `false` in the superclass. It is a client  
1003 error to specify such a qualifier in the `ModifiedClass` with a definition different than that in  
1004 the superclass (where definition encompasses the name, type, and flavor attribute settings  
1005 of the `<QUALIFIER>` XML element and the value of the qualifier).
  - 1006 – Any qualifiers defined in the superclass with a `TOSUBCLASS` attribute value of `false`  
1007 shall not be propagated to the subclass.

1008 If `ModifyClass` is successful, the WBEM server updates the specified class. The request to modify the  
1009 class shall fail if the server cannot consistently update any existing subclasses or instances of that class.

1010 If `ModifyClass` is unsuccessful, this method shall return one of the following status codes, where the error  
1011 returned is the first applicable error in the list, starting with the first element and working down. Any  
1012 additional method-specific interpretation of the error is enclosed in parentheses.

- 1013 • CIM\_ERR\_ACCESS\_DENIED
- 1014 • CIM\_ERR\_NOT\_SUPPORTED
- 1015 • CIM\_ERR\_INVALID\_NAMESPACE
- 1016 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
1017 incorrect parameters)
- 1018 • CIM\_ERR\_NOT\_FOUND (The CIM class does not exist.)
- 1019 • CIM\_ERR\_INVALID\_SUPERCLASS (The putative CIM class declares a non-existent or  
1020 incorrect superclass.)
- 1021 • CIM\_ERR\_CLASS\_HAS\_CHILDREN (The modification could not be performed because the  
1022 subclasses of the class could not be updated consistently.)
- 1023 • CIM\_ERR\_CLASS\_HAS\_INSTANCES (The modification could not be performed because the  
1024 instances of the class could not be updated consistently.)
- 1025 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 1026 5.4.2.8 ModifyInstance

1027 The ModifyInstance operation modifies an existing CIM instance in the target namespace. The instance  
1028 shall already exist:

```
1029 void ModifyInstance (
1030     [IN] <namedInstance> ModifiedInstance,
1031     [IN, OPTIONAL] boolean IncludeQualifiers = true,      (DEPRECATED)
1032     [IN, OPTIONAL, NULL] string PropertyList[] = NULL
1033 )
```

1034 The `ModifiedInstance` input parameter identifies the name of the instance to be modified and  
1035 provides the new property values.

1036 **DEPRECATION NOTE:** Use of the `IncludeQualifiers` parameter is DEPRECATED, and it may be  
1037 removed in a future version of this document. The behavior of the `IncludeQualifiers` parameter is  
1038 not specified. A WBEM client cannot rely on `IncludeQualifiers` to have any impact on the operation.  
1039 It is recommended that the WBEM server ignore any qualifiers included in `ModifiedInstance`. If the  
1040 `IncludeQualifiers` input parameter is `true`, the qualifiers are modified as specified in  
1041 `ModifiedInstance`. If the parameter is `false`, qualifiers in `ModifiedInstance` are ignored and no  
1042 qualifiers are explicitly modified.

1043 The set of properties designated to be modified shall be determined as follows:

1044 If the `PropertyList` input parameter is not `NULL`, the members of the array define one or more  
1045 property names. The properties specified in `PropertyList` are designated to be modified. Properties of  
1046 the `ModifiedInstance` that are missing from `PropertyList` are not designated to be modified. If  
1047 `PropertyList` is an empty array, no properties are designated to be modified. If `PropertyList` is  
1048 `NULL`, the properties of `ModifiedInstance` with values different from the current values in the instance  
1049 are designated to be modified.

1050 If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise  
1051 process the request normally. If `PropertyList` contains property names that are invalid for the instance  
1052 to be modified, the WBEM server shall reject the request.

1053 If a property is designated to be modified, the WBEM server shall either modify the property, or reject the  
1054 request. The server shall reject modification requests for key properties. Further considerations for  
1055 accepting or rejecting modification requests based on the properties requested to be modified are out of

1056 scope for this document; CIM model definitions are expected to cover that. Note that the WRITE qualifier  
 1057 on a property is considered to be in the area of CIM models; specifically, a value of True for the WRITE  
 1058 qualifier does not guarantee modifiability of that property, and a value of False does not prevent  
 1059 modifiability.

1060 If a property is not designated to be modified, the server shall not modify its value. However, note that  
 1061 properties may change their values as a result of other changes.

1062 In modifying the instance, the WBEM server shall conform to the following rules and ensure their  
 1063 application:

- 1064 • The server shall ignore any CLASSORIGIN and PROPAGATED attributes in the  
 1065 ModifiedInstance.
- 1066 • The class shall exist, and the client shall not change its name in the instance to be modified.
- 1067 • **DEPRECATED.** Any qualifiers in the instance not defined in the class are created as new  
 1068 elements of the instance if `IncludeQualifiers` is `true`.
- 1069 • All properties of the instance to be modified preserve their CLASSORIGIN attribute value from  
 1070 that defined in the class.
- 1071 • **DEPRECATED.** Any qualifier previously defined in the instance to be modified but not defined  
 1072 in the class, and which is not present in the `ModifiedInstance` parameter, is removed from  
 1073 the instance if `IncludeQualifiers` is `true`.
- 1074 • If a property is to be modified as previously defined, the designated new value for that property  
 1075 in the CIM instance shall be the property value (including NULL) specified in the  
 1076 `ModifiedInstance` parameter, or if the property is not specified in the `ModifiedInstance`  
 1077 parameter, the default value (including NULL) defined in the property declaration, or if the  
 1078 property does not define a default value, there is no designated new value for the property.

1079 If there is a designated new value for a property, the server shall either update the property to  
 1080 that value, or reject the request. If there is no designated new value for a property, the server  
 1081 may update the property to any value (including NULL). Further determinations about this  
 1082 decision are out of scope for this document; CIM model definitions are expected to cover that..

- 1083 • **DEPRECATION NOTE:** The use of the TOINSTANCE qualifier attribute is DEPRECATED.  
 1084 Servers may choose to ignore TOINSTANCE. Servers that do not ignore TOINSTANCE shall  
 1085 interpret it so that any qualifiers defined in the class with a TOINSTANCE attribute value of `true`  
 1086 appear in the instance. A propagated qualifier cannot be removed from an instance. qualifiers in  
 1087 the class with a TOINSTANCE attribute value of `false` shall not be propagated to the instance
- 1088 • **DEPRECATED.** Any qualifier propagated from the class cannot be modified in the instance if  
 1089 the OVERRIDABLE attribute of that qualifier is set to `false` in the class. It is a client error to  
 1090 specify such a qualifier in `ModifiedInstance` with a definition different than that in the class  
 1091 (where definition encompasses the name, type, and flavor attribute settings of the  
 1092 <QUALIFIER> XML element and the value of the qualifier).

1093 If `ModifyInstance` is successful, the specified CIM instance has been updated as described in this  
 1094 subclause.

1095 If `ModifyInstance` is unsuccessful, the specified Instance is not updated, and the method shall return one  
 1096 of the following status codes, where the error returned is the first applicable error in the list, starting with  
 1097 the first element and working down. Any additional interpretation of the error is enclosed in parentheses.

- 1098 • CIM\_ERR\_ACCESS\_DENIED
- 1099 • CIM\_ERR\_NOT\_SUPPORTED (by the WBEM server for this operation)
- 1100 • CIM\_ERR\_INVALID\_NAMESPACE

- 1101 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
1102 incorrect parameters and invalid properties to be modified)
- 1103 • CIM\_ERR\_INVALID\_CLASS (The CIM class of the instance to be modified does not exist.)
- 1104 • CIM\_ERR\_NOT\_SUPPORTED (This operation is not supported for the class of the specified  
1105 instance, if provided.)
- 1106 • CIM\_ERR\_NOT\_FOUND (The CIM instance to be modified does not exist.)
- 1107 • CIM\_ERR\_FAILED (This operation is not supported for the specified instance or some other  
1108 unspecified error occurred, including a request for non-writable properties to be modified or a  
1109 property that cannot be modified at this time.)

#### 1110 5.4.2.9 EnumerateClasses

1111 The EnumerateClasses operation enumerates subclasses of a CIM class in the target namespace:

```
1112 <class>* EnumerateClasses (
1113     [IN,OPTIONAL,NULL] <className> ClassName=NULL,
1114     [IN,OPTIONAL] boolean DeepInheritance = false,
1115     [IN,OPTIONAL] boolean LocalOnly = true,
1116     [IN,OPTIONAL] boolean IncludeQualifiers = true,
1117     [IN,OPTIONAL] boolean IncludeClassOrigin = false
1118 )
```

1119 The `ClassName` input parameter defines the class that is the basis for the enumeration.

1120 If the `DeepInheritance` input parameter is `true`, all subclasses of the specified class should be  
1121 returned. If the `ClassName` input parameter is absent, this implies that all classes in the target  
1122 namespace should be returned. If `DeepInheritance` is `false`, only immediate child subclasses are  
1123 returned. If the `ClassName` input parameter is `NULL`, this implies that all top-level classes (that is,  
1124 classes with no superclass) in the target namespace should be returned. This definition of  
1125 `DeepInheritance` applies only to the `EnumerateClasses` and `EnumerateClassName` operations.

1126 If the `LocalOnly` input parameter is `true`, any CIM elements (properties, methods, and qualifiers)  
1127 except those added or overridden in the class as specified in the `classname` input parameter shall not be  
1128 included in the returned class. If it is `false`, this parameter defines no additional filtering.

1129 If the `IncludeQualifiers` input parameter is `true`, all qualifiers for each class (including qualifiers on  
1130 the class and on any returned properties, methods, or method parameters) shall be included as  
1131 `<QUALIFIER>` XML elements in the response. If it is `false`, no `<QUALIFIER>` XML elements are  
1132 present.

1133 If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on  
1134 all appropriate elements in each returned class. If it is `false`, no `CLASSORIGIN` attributes are present.

1135 If `EnumerateClasses` is successful, the method returns zero or more classes that meet the required  
1136 criteria. These classes shall include all CIM elements (properties, methods, and qualifiers) defined in or  
1137 inherited by each class, reduced by any elements excluded as a result of using the `LocalOnly` filter.

1138 If `EnumerateClasses` is unsuccessful, this method shall return one of the following status codes, where  
1139 the error returned is the first applicable error in the list, starting with the first element and working down.  
1140 Any additional method-specific interpretation of the error is enclosed in parentheses.

- 1141 • CIM\_ERR\_ACCESS\_DENIED
- 1142 • CIM\_ERR\_NOT\_SUPPORTED

- 1143 • CIM\_ERR\_INVALID\_NAMESPACE
- 1144 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
1145 incorrect parameters)
- 1146 • CIM\_ERR\_INVALID\_CLASS (The CIM class for this enumeration does not exist.)
- 1147 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 1148 5.4.2.10 EnumerateClassNames

1149 The EnumerateClassNames operation enumerates the names of subclasses of a CIM class in the target  
1150 namespace:

```
1151 <className>* EnumerateClassNames (
1152     [IN,OPTIONAL,NULL] <className> ClassName = NULL,
1153     [IN,OPTIONAL] boolean DeepInheritance = false
1154 )
```

1155 The `ClassName` input parameter defines the class that is the basis for the enumeration.

1156 If the `DeepInheritance` input parameter is `true`, the names of all subclasses of the specified class  
1157 should be returned. If the `ClassName` input parameter is absent, this implies that the names of all classes  
1158 in the target namespace should be returned. If `DeepInheritance` is `false`, only the names of immediate  
1159 child subclasses are returned. If the `ClassName` input parameter is `NULL`, this implies that the names of  
1160 all top-level classes (that is, classes with no superclass) in the target namespace should be returned. This  
1161 definition of `DeepInheritance` applies only to the `EnumerateClasses` and `EnumerateClassName`  
1162 operations.

1163 If `EnumerateClassNames` is successful, the method returns zero or more names of classes that meet the  
1164 requested criteria.

1165 If `EnumerateClassNames` is unsuccessful, this method returns one of the following status codes, where  
1166 the error returned is the first applicable error in the list, starting with the first element and working down.  
1167 Any additional method-specific interpretation of the error is enclosed in parentheses.

- 1168 • CIM\_ERR\_ACCESS\_DENIED
- 1169 • CIM\_ERR\_NOT\_SUPPORTED
- 1170 • CIM\_ERR\_INVALID\_NAMESPACE
- 1171 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
1172 incorrect parameters)
- 1173 • CIM\_ERR\_INVALID\_CLASS (The CIM class that is the basis for this enumeration does not  
1174 exist.)
- 1175 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 1176 5.4.2.11 EnumerateInstances (DEPRECATED)

1177 The EnumerateInstances operation enumerates instances of a CIM class in the target namespace,  
1178 including instances in the class and any subclasses in accordance with the polymorphic nature of CIM  
1179 objects:

```
1180 <namedInstance>* EnumerateInstances (
1181     [IN] <className> ClassName,
1182     [IN,OPTIONAL] boolean LocalOnly = true, (DEPRECATED)
1183     [IN,OPTIONAL] boolean DeepInheritance = true,
1184     [IN,OPTIONAL] boolean IncludeQualifiers = false, (DEPRECATED)
```

```

1185         [IN,OPTIONAL] boolean IncludeClassOrigin = false,
1186         [IN,OPTIONAL,NULL] string PropertyList [] = NULL
1187     )

```

1188 **DEPRECATION NOTE:** The `EnumerateInstances` operation has been deprecated in version 1.4 of this  
 1189 document. Use `OpenEnumerateInstances` instead (see 5.4.2.24.3).

1190 The `ClassName` input parameter defines the class that is the basis for the enumeration.

1191 **DEPRECATION NOTE:** With version 1.2 of this document, the `LocalOnly` parameter is DEPRECATED.  
 1192 `LocalOnly` filtering, as defined in 1.1, will not be supported in the next major revision of this document.  
 1193 In version 1.1 of this document, the definition of the `LocalOnly` parameter was incorrectly modified. This  
 1194 change introduced a number of interoperability and backward compatibility problems for WBEM clients  
 1195 using the `LocalOnly` parameter to filter the set of properties returned. The DMTF strongly recommends  
 1196 that WBEM clients set `LocalOnly` to `false` and do not use this parameter to filter the set of properties  
 1197 returned. To minimize the impact of this recommendation on WBEM clients, a WBEM server may choose  
 1198 to treat the value of the `LocalOnly` parameter as `false` for all requests. A WBEM server shall  
 1199 consistently support a single interpretation of the `LocalOnly` parameter. Refer to ANNEX B for details.

1200 If the `DeepInheritance` input parameter is `false`, each returned instance shall not include any  
 1201 properties added by subclasses of the specified class. If it is `true`, no additional filtering is defined.

1202 **DEPRECATION NOTE:** The use of the `IncludeQualifiers` parameter is DEPRECATED and it may  
 1203 be removed in a future version of this document. The definition of `IncludeQualifiers` is ambiguous  
 1204 and when this parameter is set to `true`, WBEM clients cannot be assured that any qualifiers will be  
 1205 returned. A WBEM client should always set this parameter to `false`. To minimize the impact of this  
 1206 recommendation on WBEM clients, a WBEM server may choose to treat the value of  
 1207 `IncludeQualifiers` as `false` for all requests. The preferred behavior is to use the class operations to  
 1208 receive qualifier information and not depend on any qualifiers in this response. If the  
 1209 `IncludeQualifiers` input parameter is `true`, all qualifiers for the instance, (including qualifiers on the  
 1210 instance and on any returned properties, shall be included as `<QUALIFIER>` XML elements in the  
 1211 response. If it is `false`, no `<QUALIFIER>` XML elements are present in the returned instance.

1212 If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on  
 1213 all appropriate elements in each returned Instance. If it is `false`, no `CLASSORIGIN` attributes are  
 1214 present.

1215 If the `PropertyList` input parameter is not `NULL`, the members of the array define one or more  
 1216 property names of the designated class. This definition may include inherited property names or property  
 1217 names explicitly defined in the designated class. However, it may not include property names added in  
 1218 subclasses of the designated class. Each returned instance shall not include any properties missing from  
 1219 this list. Note that `PropertyList` acts as an additional filter on the properties defined by the `LocalOnly`  
 1220 and `DeepInheritance` input parameters; if `PropertyList` includes a property name that is not in the  
 1221 set defined by the `LocalOnly` and `DeepInheritance` combination, the element for the property shall  
 1222 not be included in the returned instances. If `PropertyList` is an empty array, no properties are included  
 1223 in the returned instances. If `PropertyList` is `NULL`, no additional filtering is defined.

1224 If `PropertyList` contains duplicate property names, the WBEM server shall ignore the duplicates but  
 1225 otherwise process the request normally. If `PropertyList` contains property names that are invalid for a  
 1226 target instance, the WBEM server shall ignore them for that instance but otherwise process the request  
 1227 normally.

1228 Properties with the `NULL` value may be omitted from the response, even if the WBEM client has not  
 1229 requested the exclusion of the property through the `LocalOnly`, `DeepInheritance`, or `PropertyList`  
 1230 filters. The WBEM client shall interpret such omitted properties as `NULL`. Note that the WBEM client

1231 cannot make any assumptions about properties omitted as a result of using any `LocalOnly`,  
1232 `DeepInheritance`, or `PropertyList` filters.

1233 If `EnumerateInstances` is successful, the method returns zero or more `<namedInstance>` items  
1234 representing named instances that meet the required criteria. These instances shall have all properties  
1235 defined in and inherited by their respective classes, reduced by any properties excluded as a result of  
1236 using the `LocalOnly`, `DeepInheritance`, or `PropertyList` filters and further reduced by any NULL-  
1237 valued properties omitted from the response.

1238 If `EnumerateInstances` is unsuccessful, this method shall return one of the following status codes, where  
1239 the error returned is the first applicable error in the list, starting with the first element and working down.  
1240 Any additional method-specific interpretation of the error is enclosed in parentheses.

- 1241 • `CIM_ERR_ACCESS_DENIED`
- 1242 • `CIM_ERR_NOT_SUPPORTED` (by the WBEM server for this operation)
- 1243 • `CIM_ERR_INVALID_NAMESPACE`
- 1244 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
1245 incorrect parameters)
- 1246 • `CIM_ERR_INVALID_CLASS` (The CIM class that is the basis for this enumeration does not  
1247 exist.)
- 1248 • `CIM_ERR_NOT_SUPPORTED` (This operation is not supported for the specified class and all  
1249 of its subclasses, if provided.)
- 1250 • `CIM_ERR_FAILED` (Some other unspecified error occurred.)

#### 1251 5.4.2.12 `EnumerateInstanceNames` (DEPRECATED)

1252 The `EnumerateInstanceNames` operation enumerates the names (model paths) of the instances of a CIM  
1253 class in the target namespace, including instances in the class and any subclasses in accordance with  
1254 the polymorphic nature of CIM objects:

```
1255     <instanceName>* EnumerateInstanceNames (
1256         [IN] <className> ClassName
1257     )
```

1258 **DEPRECATION NOTE:** The `EnumerateInstanceNames` operation has been deprecated in version 1.4 of  
1259 this document. Use `OpenEnumerateInstancePaths` instead (see 5.4.2.24.4).

1260 The `ClassName` input parameter defines the class that is the basis for the enumeration.

1261 If `EnumerateInstanceNames` is successful, the method returns zero or more `<instanceName>` items  
1262 representing instance names (referred to in [DSP0004](#) as a model path) that meet the requested criteria.  
1263 The `<instanceName>` items shall specify the class from which the instance is instantiated, not any of its  
1264 superclasses. Note that this class may be different from the class specified as input.

1265 If `EnumerateInstanceNames` is unsuccessful, this method shall return one of the following status codes,  
1266 where the error returned is the first applicable error in the list, starting with the first element and working  
1267 down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- 1268 • `CIM_ERR_ACCESS_DENIED`
- 1269 • `CIM_ERR_NOT_SUPPORTED` (by the WBEM server for this operation)
- 1270 • `CIM_ERR_INVALID_NAMESPACE`
- 1271 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
1272 incorrect parameters)

- 1273 • CIM\_ERR\_INVALID\_CLASS (The CIM class that is the basis for this enumeration does not  
1274 exist.)
- 1275 • CIM\_ERR\_NOT\_SUPPORTED (This operation is not supported for the specified class and all  
1276 of its subclasses, if provided.)
- 1277 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 1278 5.4.2.13 ExecQuery (DEPRECATED)

1279 The ExecQuery operation executes a query against the target namespace:

```
1280 <object>* ExecQuery (
1281     [IN] string QueryLanguage,
1282     [IN] string Query
1283 )
```

1284 **DEPRECATION NOTE:** The ExecQuery operation has been deprecated in version 1.4 of this document.  
1285 Use OpenQueryInstances instead (see 5.4.2.24.14).

1286 The `QueryLanguage` input parameter defines the query language in which the query parameter is  
1287 expressed.

1288 The `Query` input parameter defines the query to be executed. The results of the query shall be  
1289 constrained to contain only CIM classes that exist in the target namespace or CIM instances whose  
1290 classes exist in the target namespace. Note that any instances in the result set may or may not exist in  
1291 any namespace. Note that for query languages supporting select-lists and from-clauses, this implies that  
1292 all select-list entries resolve to disjoint properties exposed by one CIM class named in the from-clause.  
1293 This rule does not prevent such queries from using joins.

1294 Neither the query language nor the format of the query is defined by this document. It is anticipated that  
1295 query languages will be submitted to the DMTF as separate proposals.

1296 [WBEM servers](#) can declare which query languages they support (if any) using a mechanism defined in  
1297 7.5.

1298 If ExecQuery is successful, the method returns zero or more `<object>` items representing CIM classes  
1299 or instances that correspond to the results of the query.

1300 If ExecQuery is unsuccessful, the method shall return one of the following status codes, where the error  
1301 returned is the first applicable error in the list, starting with the first element and working down. Any  
1302 additional method-specific interpretation of the error is enclosed in parentheses.

- 1303 • CIM\_ERR\_ACCESS\_DENIED
- 1304 • CIM\_ERR\_NOT\_SUPPORTED
- 1305 • CIM\_ERR\_INVALID\_NAMESPACE
- 1306 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
1307 incorrect parameters)
- 1308 • CIM\_ERR\_QUERY\_LANGUAGE\_NOT\_SUPPORTED (The requested query language is not  
1309 recognized.)
- 1310 • CIM\_ERR\_INVALID\_QUERY (The query is not a valid query in the specified query language.)
- 1311 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

1312 **5.4.2.14 Associators (PARTLY DEPRECATED)**

1313 The Associators operation enumerates CIM objects (classes or instances) associated with a particular  
1314 source CIM object:

```
1315     <objectWithPath>* Associators (
1316         [IN] <objectName> ObjectName,
1317         [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
1318         [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1319         [IN,OPTIONAL,NULL] string Role = NULL,
1320         [IN,OPTIONAL,NULL] string ResultRole = NULL,
1321         [IN,OPTIONAL] boolean IncludeQualifiers = false,           (DEPRECATED)
1322         [IN,OPTIONAL] boolean IncludeClassOrigin = false,
1323         [IN,OPTIONAL,NULL] string PropertyList [] = NULL
1324     )
```

1325 **DEPRECATION NOTE:** The Associators operation for instances has been deprecated in version 1.4 of  
1326 this document. Use OpenAssociatorInstances instead (see 5.4.2.24.7). The Associators operation for  
1327 classes remains undeprecated.

1328 The `ObjectName` input parameter defines the source CIM object whose associated objects are to be  
1329 returned. This may be either a class name or instance name (model path).

1330 The `AssocClass` input parameter, if not `NULL`, shall be a valid CIM association class name. It acts as a  
1331 filter on the returned set of objects by mandating that each returned object shall be associated to the  
1332 source object through an instance of this class or one of its subclasses.

1333 The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the  
1334 returned set of objects by mandating that each returned object shall be either an instance of this class (or  
1335 one of its subclasses) or be this class (or one of its subclasses).

1336 The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned  
1337 set of objects by mandating that each returned object shall be associated with the source object through  
1338 an association in which the source object plays the specified role. That is, the name of the property in the  
1339 association class that refers to the source object shall match the value of this parameter.

1340 The `ResultRole` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the  
1341 returned set of objects by mandating that each returned object shall be associated to the source object  
1342 through an association in which the returned object plays the specified role. That is, the name of the  
1343 property in the association class that refers to the returned object shall match the value of this parameter.

1344 **DEPRECATION NOTE:** The use of the `IncludeQualifiers` parameter is DEPRECATED and it may  
1345 be removed in a future version of this document. The preferred behavior is to use the class operations to  
1346 receive qualifier information and not depend on any qualifiers in this response. If `IncludeQualifiers`  
1347 is `true`, all qualifiers for each object (including qualifiers on the object and on any returned properties)  
1348 shall be included as `<QUALIFIER>` XML elements in the response. If it is `false`, no `<QUALIFIER>` XML  
1349 elements are present.

1350 If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on  
1351 all appropriate elements in each returned object. If it is `false`, no `CLASSORIGIN` attributes are present.

1352 If the `PropertyList` input parameter is not `NULL`, the members of the array define one or more  
1353 property names. Each returned object shall not include any properties missing from this list. If  
1354 `PropertyList` is an empty array, no properties are included in each returned object. If it is `NULL`, no  
1355 additional filtering is defined.

1356 If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise  
 1357 process the request normally. If `PropertyList` contains property names that are invalid for a target  
 1358 object, the WBEM server shall ignore them for that object but otherwise process the request  
 1359 normally. Clients should not explicitly specify properties in the `PropertyList` parameter unless they  
 1360 specify a non-NULL value for the `ResultClass` parameter.

1361 If instances are returned, properties with the NULL value may be omitted from the response, even if the  
 1362 WBEM client has not requested the exclusion of the through the `PropertyList` filter. The WBEM client  
 1363 shall interpret such omitted properties as NULL. Note that the WBEM client cannot make any  
 1364 assumptions about properties omitted as a result of using the `PropertyList` filter. If classes are  
 1365 returned, the WBEM server cannot make this choice, and only the WBEM client can cause properties to  
 1366 be excluded by using the `PropertyList` filter.

1367 If `Associators` is successful, the method returns zero or more `<objectWithPath>` items representing  
 1368 CIM classes or instances meeting the requested criteria. Because it is possible for CIM objects from  
 1369 different hosts or namespaces to be associated, each returned object includes location information. If the  
 1370 `ObjectName` refers to a class, then classes are returned. These classes shall have all CIM elements  
 1371 (properties, methods, and qualifiers) defined in and inherited by that class, reduced by any properties  
 1372 excluded as a result of using the `PropertyList` filter. If the `ObjectName` refers to an instance, then  
 1373 instances are returned. These instances shall have all properties defined in and inherited by its class,  
 1374 reduced by any properties excluded as a result of using the `PropertyList` filter and further reduced by  
 1375 any NULL valued properties omitted from the response.

1376 If `Associators` is unsuccessful, this method shall return one of the following status codes, where the error  
 1377 returned is the first applicable error in the list, starting with the first element and working down. Any  
 1378 additional method-specific interpretation of the error is enclosed in parentheses.

- 1379 • CIM\_ERR\_ACCESS\_DENIED
- 1380 • CIM\_ERR\_NOT\_SUPPORTED (by the WBEM server for this operation)
- 1381 • CIM\_ERR\_INVALID\_NAMESPACE
- 1382 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
 1383 incorrect parameters)
- 1384 • CIM\_ERR\_NOT\_SUPPORTED (This operation is not supported for the class of the specified  
 1385 instance, if provided.)
- 1386 • CIM\_ERR\_FAILED (This operation is not supported for the specified instance, or some other  
 1387 unspecified error occurred.)

#### 1388 5.4.2.15 `AssociatorNames` (PARTLY DEPRECATED)

1389 The `AssociatorNames` operation enumerates the names of CIM Objects (classes or instances) that are  
 1390 associated with a particular source CIM object:

```

1391 <objectPath>* AssociatorNames (
1392     [IN] <objectName> ObjectName,
1393     [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
1394     [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1395     [IN,OPTIONAL,NULL] string Role = NULL,
1396     [IN,OPTIONAL,NULL] string ResultRole = NULL
1397 )
```

1398 **DEPRECATION NOTE:** The `AssociatorNames` operation has been deprecated in version 1.4 of this  
 1399 document. Use `OpenAssociatorInstancePaths` instead (see 5.4.2.24.8). The `AssociatorNames` operation  
 1400 for classes remains undeprecated.

- 1401 The `ObjectName` input parameter defines the source CIM object whose associated names are to be  
 1402 returned. This is either a class or instance name (model path).
- 1403 The `AssocClass` input parameter, if not `NULL`, shall be a valid CIM association class name. It acts as a  
 1404 filter on the returned set of names by mandating that each returned name identify an object that shall be  
 1405 associated to the source object through an instance of this class or one of its subclasses.
- 1406 The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the  
 1407 returned set of names by mandating that each returned name identify an object that shall be either an  
 1408 instance of this class (or one of its subclasses) or be this class (or one of its subclasses).
- 1409 The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned  
 1410 set of names by mandating that each returned name identify an object that shall be associated to the  
 1411 source object through an association in which the source object plays the specified role. That is, the  
 1412 name of the property in the association class that refers to the source object shall match the value of this  
 1413 parameter.
- 1414 The `ResultRole` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the  
 1415 returned set of names by mandating that each returned name identify an object that shall be associated  
 1416 to the source object through an association in which the named returned object plays the specified role.  
 1417 That is, the name of the property in the association class that refers to the returned object shall match the  
 1418 value of this parameter.
- 1419 If `AssociatorNames` is successful, the method returns zero or more `<objectPath>` items representing  
 1420 CIM class paths or instance paths meeting the requested criteria. Because CIM objects from different  
 1421 hosts or namespaces can be associated, each returned object includes location information. If the  
 1422 `ObjectName` refers to a class path, then class paths are returned. Otherwise, the `ObjectName` refers to  
 1423 an instance path, and instance paths are returned.
- 1424 If `AssociatorNames` is unsuccessful, one of the following status codes shall be returned by this method,  
 1425 where the first applicable error in the list (starting with the first element of the list, and working down) is  
 1426 the error returned. Any additional method-specific interpretation of the error is given in parentheses.
- 1427 • `CIM_ERR_ACCESS_DENIED`
  - 1428 • `CIM_ERR_NOT_SUPPORTED` (by the WBEM server for this operation)
  - 1429 • `CIM_ERR_INVALID_NAMESPACE`
  - 1430 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized or otherwise  
 1431 incorrect parameters)
  - 1432 • `CIM_ERR_NOT_SUPPORTED` (This operation is not supported for the class of the specified  
 1433 instance, if provided.)
  - 1434 • `CIM_ERR_FAILED` (This operation is not supported for the specified instance, or some other  
 1435 unspecified error occurred.)

#### 1436 5.4.2.16 References (PARTLY DEPRECATED)

1437 The `References` operation enumerates the association objects that refer to a particular target CIM object  
 1438 (class or instance).

```

1439 <objectWithPath>* References (
1440     [IN] <ObjectName> ObjectName,
1441     [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1442     [IN,OPTIONAL,NULL] string Role = NULL,
1443     [IN,OPTIONAL] boolean IncludeQualifiers = false,           (DEPRECATED)
1444     [IN,OPTIONAL] boolean IncludeClassOrigin = false,
```

```
1445         [IN,OPTIONAL,NULL] string PropertyList [] = NULL
1446     )
```

1447 **DEPRECATION NOTE:** The References operation has been deprecated in version 1.4 of this document.  
1448 Use OpenReferenceInstances instead (see 5.4.2.24.5). The References operation for classes remains  
1449 undeprecated.

1450 The `ObjectName` input parameter defines the target CIM object whose referring objects are to be  
1451 returned. This is either a class or instance name (model path).

1452 The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the  
1453 returned set of objects by mandating that each returned object shall be an instance of this class (or one of  
1454 its subclasses) or this class (or one of its subclasses).

1455 The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned  
1456 set of objects by mandating that each returned object shall refer to the target object through a property  
1457 with a name that matches the value of this parameter.

1458 **DEPRECATION NOTE:** The use of the `IncludeQualifiers` parameter is DEPRECATED and it may  
1459 be removed in a future version of this document. The preferred behavior is to use the class operations to  
1460 receive qualifier information and not depend on any qualifiers in this response. If `IncludeQualifiers`  
1461 is `true`, all qualifiers for each object (including qualifiers on the object and on any returned properties)  
1462 shall be included as `<QUALIFIER>` XML elements in the response. If this parameter is `false`, no  
1463 `<QUALIFIER>` XML elements are present in each returned Object.

1464 If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on  
1465 all appropriate elements in each returned object. If it is `false`, no `CLASSORIGIN` attributes are present.

1466 If the `PropertyList` input parameter is not `NULL`, the members of the array define one or more  
1467 property names. Each returned object shall not include any properties missing from this list. If  
1468 `PropertyList` is an empty array, no properties are included in each returned object. If `PropertyList`  
1469 is `NULL`, no additional filtering is defined.

1470 If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise  
1471 process the request normally. If `PropertyList` contains property names that are invalid for a target  
1472 object, the WBEM server shall ignore them for that object but otherwise process the request normally.

1473 Clients should not explicitly specify properties in the `PropertyList` parameter unless they specify a  
1474 non-`NULL` value for the `ResultClass` parameter.

1475 If instances are returned, properties with the `NULL` value may be omitted from the response, even if the  
1476 WBEM client has not requested the exclusion of the property through the `PropertyList` filter. The  
1477 WBEM client must interpret such omitted properties as `NULL`. Note that the WBEM client cannot make  
1478 any assumptions about properties omitted as a result of using the `PropertyList` filter. If classes are  
1479 returned, the WBEM server cannot make this choice, and only the WBEM client can cause properties to  
1480 be excluded by using the `PropertyList` filter.

1481 If `References` is successful, the method returns zero or more `<objectWithPath>` items representing  
1482 CIM classes or instances meeting the requested criteria. Because CIM objects from different hosts or  
1483 namespaces can be associated, each returned object includes location information. If the `ObjectName`  
1484 refers to a class, then classes are returned. These classes shall have all CIM elements (properties,  
1485 methods, and qualifiers) defined in and inherited by that class, reduced by any properties excluded as a  
1486 result of using the `PropertyList` filter. If the `ObjectName` refers to an instance, then instances are  
1487 returned. These instances shall have all properties defined in and inherited by their respective classes,  
1488 reduced by any properties excluded as a result of using the `PropertyList` filter and further reduced by  
1489 any `NULL` valued properties omitted from the response.

1490 If References is unsuccessful, this method shall return one of the following status codes, where the error  
 1491 returned is the first applicable error in the list, starting with the first element and working down. Any  
 1492 additional method-specific interpretation of the error is enclosed in parentheses.

- 1493 • CIM\_ERR\_ACCESS\_DENIED
- 1494 • CIM\_ERR\_NOT\_SUPPORTED (by the WBEM server for this operation)
- 1495 • CIM\_ERR\_INVALID\_NAMESPACE
- 1496 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
 1497 incorrect parameters)
- 1498 • CIM\_ERR\_NOT\_SUPPORTED (This operation is not supported for the class of the specified  
 1499 instance, if provided.)
- 1500 • CIM\_ERR\_FAILED (This operation is not supported for the specified instance, or some other  
 1501 unspecified error occurred.)

#### 1502 5.4.2.17 ReferenceNames (PARTLY DEPRECATED)

1503 The ReferenceNames operation enumerates the association objects that refer to a particular target CIM  
 1504 object (class or instance):

```
1505     <objectPath>* ReferenceNames (
1506         [IN] <objectName> ObjectName,
1507         [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1508         [IN,OPTIONAL,NULL] string Role = NULL
1509     )
```

1510 **DEPRECATION NOTE:** The ReferenceNames operation has been deprecated in version 1.4 of this  
 1511 document. Use OpenReferenceInstancePaths instead (see 5.4.2.24.6). The ReferenceNames operation  
 1512 for classes remains undeprecated.

1513 The ObjectName input parameter defines the target CIM object with the referring object names to be  
 1514 returned. It may be either a class or an instance name (model path).

1515 The ResultClass input parameter, if not NULL, shall be a valid CIM class name. It acts as a filter on the  
 1516 returned set of object names by mandating that each returned Object Name identify an instance of this  
 1517 class (or one of its subclasses) or this class (or one of its subclasses).

1518 The Role input parameter, if not NULL, shall be a valid property name. It acts as a filter on the returned  
 1519 set of object names by mandating that each returned object name shall identify an object that refers to the  
 1520 target instance through a property with a name that matches the value of this parameter.

1521 If ReferenceNames is successful, the method returns zero or more <objectPath> items representing  
 1522 CIM class paths or instance paths meeting the requested criteria. Because CIM objects from different  
 1523 hosts or namespaces can be associated, each returned object includes location information. If the  
 1524 ObjectName refers to a class path, then class paths are returned. Otherwise, the ObjectName refers to  
 1525 an instance path, and instance paths are returned.

1526 If ReferenceNames is unsuccessful, this method shall return one of the following status codes, where the  
 1527 error returned is the first applicable error in the list, starting with the first element and working down. Any  
 1528 additional method-specific interpretation of the error is enclosed in parentheses.

- 1529 • CIM\_ERR\_ACCESS\_DENIED
- 1530 • CIM\_ERR\_NOT\_SUPPORTED (by the WBEM server for this operation)
- 1531 • CIM\_ERR\_INVALID\_NAMESPACE

- 1532 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
1533 incorrect parameters)
- 1534 • CIM\_ERR\_NOT\_SUPPORTED (This operation is not supported for the class of the specified  
1535 instance, if provided.)
- 1536 • CIM\_ERR\_FAILED (This operation is not supported for the specified instance, or some other  
1537 unspecified error occurred.)

#### 1538 5.4.2.18 GetProperty (DEPRECATED)

1539 The GetProperty operation retrieves a single property value from a CIM instance in the target  
1540 namespace:

```
1541     <propertyValue> GetProperty (
1542         [IN] <instanceName> InstanceName,
1543         [IN] string PropertyName
1544     )
```

1545 **DEPRECATION NOTE:** The GetProperty operation has been deprecated in version 1.4 of this document.  
1546 Use GetInstance instead (see 5.4.2.2).

1547 The InstanceName input parameter specifies the name of the instance (model path) from which the  
1548 property value is requested.

1549 The PropertyName input parameter specifies the name of the property with the value to be returned.

1550 If GetProperty is successful, the return value specifies the value of the requested property. If the value is  
1551 NULL, no element is returned.

1552 If GetProperty is unsuccessful, this method shall return one of the following status codes, where the error  
1553 returned is the first applicable error in the list, starting with the first element and working down. Any  
1554 additional method-specific interpretation of the error is enclosed in parentheses.

- 1555 • CIM\_ERR\_ACCESS\_DENIED
- 1556 • CIM\_ERR\_INVALID\_NAMESPACE
- 1557 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
1558 incorrect parameters)
- 1559 • CIM\_ERR\_INVALID\_CLASS (The CIM class does not exist in the specified namespace.)
- 1560 • CIM\_ERR\_NOT\_FOUND (The CIM class exists, but the requested CIM instance does not exist  
1561 in the specified namespace.)
- 1562 • CIM\_ERR\_NO\_SUCH\_PROPERTY (The CIM instance exists, but the requested property does  
1563 not.)
- 1564 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 1565 5.4.2.19 SetProperty (DEPRECATED)

1566 The SetProperty operation sets a single property value in a CIM instance in the target namespace:

```
1567     void SetProperty (
1568         [IN] <instanceName> InstanceName,
1569         [IN] string PropertyName,
1570         [IN,OPTIONAL,NULL] <propertyValue> NewValue = NULL
1571     )
```

1572 **DEPRECATION NOTE:** The SetProperty operation has been deprecated in version 1.4 of this document.  
 1573 Use ModifyInstance instead (see 5.4.2.8).

1574 The InstanceName input parameter specifies the name of the instance (model path) with the property  
 1575 value to be updated.

1576 The PropertyName input parameter specifies the name of the property with the value to be updated.

1577 The NewValue input parameter specifies the new value for the property (which may be NULL).

1578 If SetProperty is unsuccessful, this method shall return one of the following status codes, where the error  
 1579 returned is the first applicable error in the list, starting with the first element and working down. Any  
 1580 additional method-specific interpretation of the error is enclosed in parentheses.

- 1581 • CIM\_ERR\_ACCESS\_DENIED
- 1582 • CIM\_ERR\_NOT\_SUPPORTED (by the WBEM server for this operation)
- 1583 • CIM\_ERR\_INVALID\_NAMESPACE
- 1584 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
 1585 incorrect parameters)
- 1586 • CIM\_ERR\_INVALID\_CLASS (The CIM class does not exist in the specified namespace.)
- 1587 • CIM\_ERR\_NOT\_FOUND (The CIM class exists, but the requested CIM instance does not exist  
 1588 in the specified namespace.)
- 1589 • CIM\_ERR\_NOT\_SUPPORTED (This operation is not supported for the class of the specified  
 1590 instance, if provided.)
- 1591 • CIM\_ERR\_NO\_SUCH\_PROPERTY (The CIM instance exists, but the requested property does  
 1592 not.)
- 1593 • CIM\_ERR\_TYPE\_MISMATCH (The supplied value is incompatible with the type of the  
 1594 property.)
- 1595 • CIM\_ERR\_FAILED (This operation is not supported for the specified instance, or some other  
 1596 unspecified error occurred.)

#### 1597 5.4.2.20 GetQualifier

1598 The GetQualifier operation retrieves a single qualifier declaration from the target namespace.

```
1599     <qualifierDecl> GetQualifier (
1600         [IN] string QualifierName
1601     )
```

1602 The QualifierName input parameter identifies the qualifier with the declaration to be retrieved.

1603 If GetQualifier is successful, the method returns the qualifier declaration for the named qualifier.

1604 If GetQualifier is unsuccessful, this method shall return one of the following status codes, where the error  
 1605 returned is the first applicable error in the list, starting with the first element and working down. Any  
 1606 additional method-specific interpretation of the error is enclosed in parentheses.

- 1607 • CIM\_ERR\_ACCESS\_DENIED
- 1608 • CIM\_ERR\_NOT\_SUPPORTED
- 1609 • CIM\_ERR\_INVALID\_NAMESPACE

- 1610 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise
- 1611 incorrect parameters)
- 1612 • CIM\_ERR\_NOT\_FOUND (The requested qualifier declaration does not exist.)
- 1613 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 1614 5.4.2.21 SetQualifier

1615 The SetQualifier operation creates or updates a single qualifier declaration in the target namespace. If the  
1616 qualifier declaration already exists, it is overwritten:

```
1617 void SetQualifier (
1618     [IN] <qualifierDecl> QualifierDeclaration
1619 )
```

1620 The `QualifierDeclaration` input parameter defines the qualifier declaration to add to the  
1621 namespace.

1622 If SetQualifier is successful, the qualifier declaration is added to the target namespace. If a qualifier  
1623 declaration with the same qualifier name already exists, the new declaration replaces it.

1624 If SetQualifier is unsuccessful, this method returns one of the following status codes, where the error  
1625 returned is the first applicable error in the list, starting with the first element and working down. Any  
1626 additional method-specific interpretation of the error is enclosed in parentheses.

- 1627 • CIM\_ERR\_ACCESS\_DENIED
- 1628 • CIM\_ERR\_NOT\_SUPPORTED
- 1629 • CIM\_ERR\_INVALID\_NAMESPACE
- 1630 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise
- 1631 incorrect parameters)
- 1632 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 1633 5.4.2.22 DeleteQualifier

1634 The DeleteQualifier operation deletes a single qualifier declaration from the target namespace.

```
1635 void DeleteQualifier (
1636     [IN] string QualifierName
1637 )
```

1638 The `QualifierName` input parameter identifies the qualifier with the declaration to be deleted.

1639 If DeleteQualifier is successful, the specified qualifier declaration is deleted from the namespace.

1640 If DeleteQualifier is unsuccessful, this method shall return one of the following status codes, where the  
1641 error returned is the first applicable error in the list, starting with the first element and working down. Any  
1642 additional method-specific interpretation of the error is enclosed in parentheses.

- 1643 • CIM\_ERR\_ACCESS\_DENIED
- 1644 • CIM\_ERR\_NOT\_SUPPORTED
- 1645 • CIM\_ERR\_INVALID\_NAMESPACE
- 1646 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise
- 1647 incorrect parameters)
- 1648 • CIM\_ERR\_NOT\_FOUND (The requested qualifier declaration does not exist.)

- 1649 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 1650 5.4.2.23 EnumerateQualifiers

1651 The EnumerateQualifiers operation enumerates qualifier declarations from the target namespace.

```
1652 <qualifierDecl>* EnumerateQualifiers (
1653 )
```

1654 If EnumerateQualifiers is successful, the method returns zero or more <qualifierDecl> items  
1655 representing qualifier declarations.

1656 If EnumerateQualifiers is unsuccessful, this method shall return one of the following status codes, where  
1657 the error returned is the first applicable error in the list, starting with the first element and working down.  
1658 Any additional method-specific interpretation of the error is enclosed in parentheses.

- 1659 • CIM\_ERR\_ACCESS\_DENIED
- 1660 • CIM\_ERR\_NOT\_SUPPORTED
- 1661 • CIM\_ERR\_INVALID\_NAMESPACE
- 1662 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
1663 incorrect parameters)
- 1664 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 1665 5.4.2.24 Pulled Enumeration Operations

1666 This clause defines a set of operations that return CIM instances or instance paths in portions controlled  
1667 by the WBEM client. These operations are called *pulled enumerations*. Usually, an enumeration session  
1668 is established through an Open operation, and subsequent repeated executions of a Pull operation on the  
1669 enumeration session are used to retrieve them. Optionally, the Open operation can also pull a first set of  
1670 items.

1671 Pulled enumeration operations consist of the following individual operations:

- 1672 • Open operations open an enumeration of the following instances or instance paths:
  - 1673 – OpenEnumerateInstances (instances of a class)
  - 1674 – OpenEnumerateInstancePaths (instance paths of instances of a class)
  - 1675 – OpenReferenceInstances (association instances referencing a target instance)
  - 1676 – OpenReferenceInstancePaths (the instance paths of association instances referencing a  
1677 target instance)
  - 1678 – OpenAssociatorInstances (instances associated with a source instance)
  - 1679 – OpenAssociatorInstancePaths (the instance paths of instances associated to a source  
1680 instance)
  - 1681 – OpenQueryInstances (the rows resulting from a query)
- 1682 • Pull operations are for the following cases:
  - 1683 – PullInstances (Instances are enumerated, and instance paths are either not available, for  
1684 example as in for OpenQueryInstances, or not desired.)
  - 1685 – PullInstancesWithPath (Instances with paths are enumerated.)
  - 1686 – PullInstancePaths (Instance paths are enumerated.)

- 1687 • Other operations are as follows:
- 1688 – CloseEnumeration (closes an open enumeration)
- 1689 – EnumerationCount (estimates the number of items in an open enumeration)

#### 1690 5.4.2.24.1 Behavioral Rules for Pulled Enumeration Operations

1691 A central concept of pulled enumeration operations is the "enumeration session," which provides a  
1692 context in which the operations perform their work and which determines the set of instances or instance  
1693 paths to be returned. To process the operations of an enumeration session, some parameters of the  
1694 Open operation need to be maintained as long as the enumeration session is open. In addition, some  
1695 state data about where the enumeration session is with regard to instances or instance paths already  
1696 returned must be maintained.

1697 From a WBEM client perspective, an enumeration session is an enumeration context value. A successful  
1698 Open operation establishes the enumeration session and returns an enumeration context value  
1699 representing it. This value is used as an input/output parameter in subsequent Pull operations on that  
1700 enumeration session. The enumeration context value shall uniquely identify the open enumeration  
1701 session within the target CIM namespace of the Open operation that established the enumeration  
1702 session. It is valid for a WBEM server to use NULL as an enumeration context value representing a  
1703 closed enumeration session, but a WBEM client shall not rely on that.

1704 Defining the enumeration context value in Pull operations as both an input parameter and an output  
1705 parameter allows the WBEM server to change the enumeration context value during the execution of a  
1706 pull operation. This ability to change allows different implementation approaches on the WBEM server  
1707 side, which are transparent for the WBEM client. Example approaches are as follows:

- 1708 • Maintain any state data describing the enumeration session internally in the WBEM server. The  
1709 enumeration context value does not need to change in subsequent Pull operations. The WBEM  
1710 server uses this value only to identify the internal state data for the open enumeration session. It  
1711 does not use the value to store any state data. A variation of this approach is to hand back  
1712 modified enumeration context values for additional WBEM server-side sequence checking.
- 1713 • Maintain any state data describing the enumeration session only on the WBEM client side. All  
1714 state data is stored in the enumeration context value, and the WBEM server does not maintain  
1715 any state data about the enumeration session, essentially being completely stateless with  
1716 regard to the enumeration session.
- 1717 • A combination of the two previous approaches.

1718 A WBEM server may support keeping enumeration sessions open across connection terminations and  
1719 shutdowns of the server. Objects may be created, deleted, or modified concurrently with an enumeration  
1720 session that involves these objects. Such changes may or may not be reflected in the enumeration set.  
1721 Therefore, there is no guarantee to the WBEM client that the enumeration set represents a consistent  
1722 snapshot of its instances at a point in time. However, the WBEM server should make a best effort attempt  
1723 for the returned enumeration set to represent a consistent snapshot of its instances at a point in time. The  
1724 order of instances in the enumeration set is undefined.

1725 This document does not restrict the number of enumeration sessions that can be established or executed  
1726 concurrently in the same WBEM server or client. This remains true even if the enumeration sets of such  
1727 concurrently established enumeration sessions contain the same instances.

1728 Except for CloseEnumeration, all operations on a particular enumeration session shall be executed  
1729 sequentially. An enumeration session can be open or closed. It is considered open if operations using its  
1730 enumeration context value as an input parameter can be executed successfully. It is opened by the  
1731 successful completion of an Open operation and closed by one of the following events:

- 1732 • Successful completion of a CloseEnumeration operation

- 1733 • Successful completion of an open or pull operation with the `EndOfSequence` output parameter  
1734 set to `true`
- 1735 • Unsuccessful completion of a pull operation when `ContinueOnError` is not requested
- 1736 • WBEM server-side decision to close the enumeration session based upon an operation timeout
- 1737 • WBEM server-side decision to close an enumeration session during an operation on that  
1738 enumeration session based upon exceeding server limits

1739 A conformant WBEM server may support closure of enumeration sessions based upon exceeding server  
1740 limits. Example situations for such a decision are:

- 1741 • Pull operations with no objects requested that are repeated with a high frequency on the same  
1742 enumeration session
- 1743 • EnumerationCount operations repeated with a high frequency on the same enumeration  
1744 session

1745 A mechanism by which WBEM servers can declare support for closure of enumeration sessions based  
1746 upon exceeding server limits is defined in 7.5. If a WBEM server supports such closure of enumeration  
1747 sessions, it shall make the decision to close during an operation on that enumeration session. There is no  
1748 way to indicate the reason for the closure if the decision is made elsewhere. If a WBEM server closes an  
1749 enumeration session based upon exceeding server limits, it shall return failure on the operation on that  
1750 enumeration session with the status code [CIM\\_ERR\\_SERVER\\_LIMITS\\_EXCEEDED](#).

#### 1751 5.4.2.24.2 Common Parameters for the Open Operations

1752 This clause defines commonly used parameters for the Open operations. The description of the individual  
1753 Open operations references these parameters as appropriate. Note that not every Open operation uses  
1754 every one of these common parameters:

- 1755 • `EnumerationContext`
  - 1756 – This output parameter is the enumeration context value representing the enumeration  
1757 session. If the `EndOfSequence` is `true`, the `EnumerationContext` value may be `NULL`.
  - 1758 – The representation of an enumeration context value uses a string type. In version 1.3 of  
1759 this document, enumeration context values were represented using the  
1760 `ENUMERATIONCONTEXT` XML element. The representation was changed to using a  
1761 string type in version 1.4 of this document, because it had turned out that all known  
1762 implementations had implemented the enumeration context value using a string type.
- 1763 • `EndOfSequence`
  - 1764 – This output parameter indicates to the WBEM client whether the enumeration session is  
1765 exhausted. If `EndOfSequence` is `true` upon successful completion of an Open operation,  
1766 no more instances are available and the WBEM server closes the enumeration session,  
1767 releasing any allocated resources related to the enumeration session. If the enumeration  
1768 set is empty, it is valid for a WBEM server to set `EndOfSequence` to `true`, even if  
1769 `MaxObjectCount` is 0. In this case, the enumeration session is closed upon successful  
1770 completion of the Open operation. If `EndOfSequence` is `false`, additional instances may  
1771 be available and the WBEM server shall not close the enumeration session.
- 1772 • `IncludeClassOrigin`
  - 1773 – This input parameter is used only on Open operations that enumerate CIM instances. It  
1774 controls whether information about the class origin of properties, references or methods is  
1775 included in any enumerated CIM instances. If `IncludeClassOrigin` is `true`, the  
1776 `CLASSORIGIN` attribute shall be present on all appropriate elements in each CIM instance  
1777 returned by any subsequent PullInstance operations on this enumeration session. If

- 1778            `IncludeClassOrigin` is false, any CLASSORIGIN attributes shall not be present in  
1779            any enumerated instances.
- 1780            • `FilterQueryLanguage` and `FilterQuery`
- 1781            – These input parameters specify a filter query that acts as an additional restricting filter on  
1782            the set of enumerated instances.
- 1783            – WBEM servers shall support filter queries in pulled enumerations and shall support the  
1784            DMTF Filter Query Language (FQL, see [DSP0212](#)) as a query language for such filter  
1785            queries. WBEM servers may support additional query languages for pulled enumerations.  
1786            A mechanism by which WBEM servers can declare the query languages they support for  
1787            pulled enumerations is not defined in this document; it is anticipated that a CIM model  
1788            based approach for declaring supported query languages is developed.
- 1789            Note that before version 1.4 of this document, support for filter queries in pulled  
1790            enumerations was optional and no particular query language was required. As a  
1791            consequence of this change, the status code  
1792            `CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED` is no longer used in CIM-  
1793            XML.
- 1794            – If `FilterQueryLanguage` is not NULL, it shall specify a query language and  
1795            `FilterQuery` shall be a valid query in that query language.
- 1796            If the query language specified in `FilterQueryLanguage` is not supported by the WBEM  
1797            server, it shall return an error with status code  
1798            `CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED`.
- 1799            If the query language specified in `FilterQueryLanguage` is supported by the WBEM  
1800            server, it shall process the filter query specified by the `FilterQuery` and  
1801            `FilterQueryLanguage` parameters, and shall either restrict the set of enumerated  
1802            instances as specified by the query language, or return an error.
- 1803            WBEM servers shall support the Filter Query Language (see [DSP0212](#)) as a query  
1804            language for pulled enumerations. WBEM servers may support additional query languages  
1805            for pulled enumerations.
- 1806            – The query specified in `FilterQuery` shall conform to the following:
- 1807            • If the query language supports specifying a set of classes the query applies to (for  
1808            example, CQL in its FROM list), only the class named in the `ClassName` parameter  
1809            shall be specified.
- 1810            • If the query language supports specifying a result list (for example, CQL in its  
1811            SELECT list), a result list may be specified in the query, but the result list shall be  
1812            ignored.
- 1813            • The query shall not define any ordering criteria or any grouping of objects.
- 1814            If the query does not satisfy these rules or if the query is invalid according to the definition  
1815            of the query language, the WBEM server shall return an error with status code  
1816            `CIM_ERR_INVALID_QUERY`. The Filter Query Language (see [DSP0212](#)) automatically  
1817            satisfies these rules.
- 1818            • `OperationTimeout`
- 1819            – This input parameter determines the minimum time the WBEM server shall maintain the  
1820            open enumeration session after the last Open or Pull operation (unless the enumeration  
1821            session is closed during the last operation). If the operation timeout is exceeded, the  
1822            WBEM server may close the enumeration session at any time, releasing any resources  
1823            allocated to the enumeration session.

- 1824 – An `OperationTimeout` of 0 means that there is no operation timeout. That is, the  
1825 enumeration session is never closed based on time.
- 1826 – If `OperationTimeout` is NULL, the WBEM server shall choose an operation timeout.
- 1827 – All other values for `OperationTimeout` specify the operation timeout in seconds.
- 1828 – A WBEM server may restrict the set of allowable values for `OperationTimeout`.  
1829 Specifically, the WBEM server may not allow 0 (no timeout). If the specified value is not an  
1830 allowable value, the WBEM server shall return failure with the status code  
1831 `CIM_ERR_INVALID_OPERATION_TIMEOUT`. A mechanism by which WBEM servers can  
1832 declare the allowable values for `OperationTimeout` is defined in 7.5.
- 1833 • `ContinueOnError`
    - 1834 – This input parameter, if true, requests a continuation on error, which is the ability to  
1835 resume an enumeration session successfully after a Pull operation returns an error. A  
1836 mechanism by which conformant WBEM servers can declare support for continuation on  
1837 error is defined in 7.5.
    - 1838 – If a WBEM server does not support continuation on error and `ContinueOnError` is true,  
1839 it shall return a failure with the status code  
1840 [CIM\\_ERR\\_CONTINUATION\\_ON\\_ERROR\\_NOT\\_SUPPORTED](#).
    - 1841 – If a WBEM server supports continuation on error and `ContinueOnError` is true, the  
1842 enumeration session shall remain open when a Pull operation fails, and any subsequent  
1843 successful Pull operations shall return the set of instances or instance paths that would  
1844 have been returned if the failing Pull operations were successful. This behavior is subject  
1845 to the consistency rules defined for pulled enumerations. If `ContinueOnError` is false,  
1846 the enumeration session shall be closed when a Pull operation returns a failure.
  - 1847 • `MaxObjectCount`
    - 1848 – This input parameter defines the maximum number of instances or instance paths that this  
1849 Open operation can return. Any uint32 number is valid, including 0. The WBEM server may  
1850 deliver any number of instances or instance paths up to `MaxObjectCount` but shall not  
1851 deliver more than `MaxObjectCount` elements. A conformant WBEM server  
1852 implementation may choose to never return any instances or instance paths during an  
1853 Open operation, regardless of the value of `MaxObjectCount`. Note that a WBEM client  
1854 can use a `MaxObjectCount` value of 0 to specify that it does not want to retrieve any  
1855 instances in the Open operation.
  - 1856 • Return Value (array of enumerated elements)
    - 1857 – The return value of a successful Open operation is an array of enumerated elements with a  
1858 number of entries from 0 up to a maximum defined by `MaxObjectCount`. These entries  
1859 meet the criteria defined in the Open operation. Note that returning no entries in the array  
1860 does not imply that the enumeration session is exhausted. Only the `EndOfSequence`  
1861 output parameter indicates whether the enumeration session is exhausted.

#### 1862 5.4.2.24.3 OpenEnumerateInstances

1863 The `OpenEnumerateInstances` operation establishes and opens an enumeration session of the instances  
1864 of a CIM class (including instances of its subclasses) in the target namespace. Optionally, it retrieves a  
1865 first set of instances.

```
1866 <instanceWithPath>* OpenEnumerateInstances (
1867     [OUT] string EnumerationContext,
1868     [OUT] Boolean EndOfSequence,
1869     [IN] <className> ClassName,
```

```

1870     [IN,OPTIONAL] boolean DeepInheritance = true,
1871     [IN,OPTIONAL] boolean IncludeClassOrigin = false,
1872     [IN,OPTIONAL,NULL] string PropertyList [] = NULL,
1873     [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
1874     [IN,OPTIONAL,NULL] string FilterQuery = NULL,
1875     [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
1876     [IN,OPTIONAL] Boolean ContinueOnError = false,
1877     [IN,OPTIONAL] uint32 MaxObjectCount = 0
1878 )

```

1879 The `OpenEnumerateInstances` operation shall comply with the behavior defined in 5.4.2.24.1.

1880 The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

1881 The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

1882 The `ClassName` input parameter defines the class that is the basis for the enumeration. The enumeration set shall consist of all instances of that specified class, including any instances of any of its subclasses, in accordance with the polymorphic nature of CIM objects.

1885 The `DeepInheritance` input parameter acts as a filter on the properties included in any enumerated CIM instances. If the `DeepInheritance` input parameter is `true`, all properties of each enumerated instance of the class shall be present (subject to constraints imposed by the other parameters), including any added by subclassing the specified class. If `DeepInheritance` is `false`, each enumerated instance includes only properties defined for the class specified by `ClassName`.

1890 The `IncludeClassOrigin` input parameter is defined in 5.4.2.24.2.

1891 The `PropertyList` input parameter acts as a filter on the properties in any enumerated CIM instances. If `PropertyList` is not `NULL`, the members of the array define zero or more property names of the specified class. This array may include inherited property names or property names explicitly defined in the specified class. However, it shall not include property names defined in subclasses of the specified class. Each enumerated instance shall not include any properties missing from this list. Note that `PropertyList` acts as an additional filter on the properties defined by the `DeepInheritance` input parameter. If `PropertyList` includes a property that is not in the set defined by `DeepInheritance`, the element for the property shall not be included. If `PropertyList` is an empty array, no properties are included in the enumerated instances. If `PropertyList` is `NULL`, no additional filtering is defined. If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise process the request normally. If `PropertyList` contains property names that are invalid for a target instance, the WBEM server shall ignore them for that instance but otherwise process the request normally.

1904 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

1905 The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

1906 The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

1907 The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

1908 If `OpenEnumerateInstances` is successful, the return value shall be an array of `<instanceWithPath>` items representing enumerated instances as defined in 5.4.2.24.2.

1910 The `PullInstancesWithPath` operation shall be used to pull instances for an enumeration session opened using `OpenEnumerateInstances`. If any other operation is used to pull instances, the WBEM server shall return failure with the status code `CIM_ERR_FAILED`.

1913 If `OpenEnumerateInstances` is unsuccessful, this operation shall return one of the following status codes,  
 1914 where the error returned is the first applicable error in the list, starting with the first element and working  
 1915 down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- 1916 • `CIM_ERR_ACCESS_DENIED`
- 1917 • `CIM_ERR_SERVER_IS_SHUTTING_DOWN`
- 1918 • `CIM_ERR_NOT_SUPPORTED`
- 1919 • `CIM_ERR_INVALID_NAMESPACE`
- 1920 • `CIM_ERR_INVALID_OPERATION_TIMEOUT`
- 1921 • `CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED`
- 1922 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
 1923 incorrect parameters)
- 1924 • `CIM_ERR_INVALID_CLASS` (The CIM class that is the basis for this enumeration does not  
 1925 exist.)
- 1926 • `CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED`
- 1927 • `CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED` (The requested filter query language is  
 1928 not recognized.)
- 1929 • `CIM_ERR_INVALID_QUERY` (The filter query is not a valid query in the specified filter query  
 1930 language.)
- 1931 • `CIM_ERR_FAILED` (Some other unspecified error occurred.)

#### 1932 5.4.2.24.4 `OpenEnumerateInstancePaths`

1933 The `OpenEnumerateInstancePaths` operation establishes and opens an enumeration session of the  
 1934 instance paths of the instances of a CIM class (including instances of its subclasses) in the target  
 1935 namespace. Optionally, it retrieves a first set of instance paths:

```

1936 <instancePath>* OpenEnumerateInstancePaths (
1937     [OUT] string EnumerationContext,
1938     [OUT] Boolean EndOfSequence,
1939     [IN] <className> ClassName,
1940     [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
1941     [IN,OPTIONAL,NULL] string FilterQuery = NULL,
1942     [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
1943     [IN,OPTIONAL] Boolean ContinueOnError = false,
1944     [IN,OPTIONAL] uint32 MaxObjectCount = 0
1945 )
  
```

1946 The `OpenEnumerateInstancePaths` operation shall comply with the behavior defined in 5.4.2.24.1.

1947 The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

1948 The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

1949 The `ClassName` input parameter defines the class that is the basis for the enumeration. The  
 1950 enumeration set shall consist of the instance paths of all instances of the specified class, including any  
 1951 instances of any of its subclasses, in accordance with the polymorphic nature of CIM objects.

1952 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

1953 The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

- 1954 The `ContinueOnError` input parameter is defined in 5.4.2.24.2.
- 1955 The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.
- 1956 If `OpenEnumerateInstancePaths` is successful, the return value shall be an array of `<instancePath>`  
1957 items representing enumerated instance paths as defined in 5.4.2.24.2.
- 1958 The `PullInstancePaths` operation shall be used to pull instances for an enumeration session opened using  
1959 `OpenEnumerateInstancePaths`. If any other operation is used to pull instances, the WBEM server shall  
1960 return failure with the status code `CIM_ERR_FAILED`.
- 1961 If `OpenEnumerateInstancePaths` is unsuccessful, this operation shall return one of the following status  
1962 codes, where the error returned is the first applicable error in the list, starting with the first element and  
1963 working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.
- 1964 • `CIM_ERR_ACCESS_DENIED`
  - 1965 • `CIM_ERR_SERVER_IS_SHUTTING_DOWN`
  - 1966 • `CIM_ERR_NOT_SUPPORTED`
  - 1967 • `CIM_ERR_INVALID_NAMESPACE`
  - 1968 • `CIM_ERR_INVALID_OPERATION_TIMEOUT`
  - 1969 • `CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED`
  - 1970 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
1971 incorrect parameters)
  - 1972 • `CIM_ERR_INVALID_CLASS` (The CIM class that is the basis for this enumeration does not  
1973 exist.)
  - 1974 • `CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED`
  - 1975 • `CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED` (The requested filter query language is  
1976 not recognized.)
  - 1977 • `CIM_ERR_INVALID_QUERY` (The filter query is not a valid query in the specified filter query  
1978 language.)
  - 1979 • `CIM_ERR_FAILED` (Some other unspecified error occurred.)

#### 1980 5.4.2.24.5 OpenReferenceInstances

1981 The `OpenReferenceInstances` operation establishes and opens the enumeration session of association  
1982 instances that refer to a particular target CIM instance in the target namespace. Optionally, it retrieves a  
1983 first set of instances:

```
1984 <instanceWithPath>* OpenReferenceInstances (
1985     [OUT] string EnumerationContext,
1986     [OUT] Boolean EndOfSequence,
1987     [IN] <instanceName> InstanceName,
1988     [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1989     [IN,OPTIONAL,NULL] string Role = NULL,
1990     [IN,OPTIONAL] boolean IncludeClassOrigin = false,
1991     [IN,OPTIONAL,NULL] string PropertyList [] = NULL,
1992     [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
1993     [IN,OPTIONAL,NULL] string FilterQuery = NULL,
1994     [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
1995     [IN,OPTIONAL] Boolean ContinueOnError = false,
```

```
1996         [IN,OPTIONAL] uint32 MaxObjectCount = 0
1997     )
```

1998 The `OpenReferenceInstances` operation shall comply with the behavior defined in 5.4.2.24.1.

1999 The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

2000 The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

2001 The `InstanceName` input parameter specifies an instance name (model path) that identifies the target  
2002 CIM instance with the referring association instances to be enumerated. Unless restricted by any of the  
2003 filter parameters of this operation, the enumeration set shall consist of all association instances that  
2004 reference the target instance.

2005 The `ResultClass` input parameter, if not `NULL`, shall be a CIM class name. It acts as a filter on the  
2006 enumerated set of instances by mandating that each enumerated instance shall be an instance of this  
2007 class or one of its subclasses. The WBEM server shall not return an error if the `ResultClass` input  
2008 parameter value is an invalid class name or if the class does not exist in the target namespace,

2009 The `Role` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the enumerated set  
2010 of instances by mandating that each enumerated instance shall refer to the target instance through a  
2011 property with a name that matches the value of this parameter. The WBEM server shall not return an  
2012 error if the `Role` input parameter value is an invalid property name or if the property does not exist,

2013 The `IncludeClassOrigin` input parameter is defined in 5.4.2.24.2.

2014 The `PropertyList` input parameter acts as a filter on the properties included in any enumerated CIM  
2015 instances. If `PropertyList` is not `NULL`, the members of the array define zero or more property names.  
2016 Each enumerated instance shall not include any properties missing from this list. If `PropertyList` is an  
2017 empty array, no properties are included in each enumerated instance. If `PropertyList` is `NULL`, all  
2018 properties are included in each enumerated instance, subject to the conditions expressed by the other  
2019 parameters. If `PropertyList` contains duplicate property names, the WBEM server shall ignore them  
2020 but otherwise process the request normally. If `PropertyList` contains property names that are invalid  
2021 for a target instance, the WBEM server shall ignore them for that instance but otherwise process the  
2022 request normally. WBEM clients should not specify properties in `PropertyList` unless they specify a  
2023 non-`NULL` value for the `ResultClass` parameter.

2024 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

2025 The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

2026 The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

2027 The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

2028 If `OpenReferenceInstances` is successful, the return value shall be an array of `<instanceWithPath>`  
2029 items representing enumerated instances as defined in 5.4.2.24.2.

2030 The `PullInstancesWithPath` operation shall be used to pull instances for an enumeration session opened  
2031 using `OpenReferenceInstances`. If any other operation is used to pull instances, the WBEM server shall  
2032 return failure with the status code `CIM_ERR_FAILED`.

2033 If `OpenReferenceInstances` is unsuccessful, this operation shall return one of the following status codes,  
2034 where the error returned is the first applicable error in the list, starting with the first element of and working  
2035 down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- 2036 • `CIM_ERR_ACCESS_DENIED`

- 2037 • CIM\_ERR\_SERVER\_IS\_SHUTTING\_DOWN
- 2038 • CIM\_ERR\_NOT\_SUPPORTED
- 2039 • CIM\_ERR\_INVALID\_NAMESPACE
- 2040 • CIM\_ERR\_INVALID\_OPERATION\_TIMEOUT
- 2041 • CIM\_ERR\_CONTINUATION\_ON\_ERROR\_NOT\_SUPPORTED
- 2042 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized or otherwise
- 2043 incorrect parameters)
- 2044 • CIM\_ERR\_NOT\_FOUND (The target instance was not found.)
- 2045 • CIM\_ERR\_FILTERED\_ENUMERATION\_NOT\_SUPPORTED
- 2046 • CIM\_ERR\_QUERY\_LANGUAGE\_NOT\_SUPPORTED (The requested filter query language is
- 2047 not recognized.)
- 2048 • CIM\_ERR\_INVALID\_QUERY (The filter query is not a valid query in the specified filter query
- 2049 language.)
- 2050 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 2051 5.4.2.24.6 OpenReferenceInstancePaths

2052 The OpenReferenceInstancePaths operation establishes and opens an enumeration session of the  
 2053 instance paths of the association instances that refer to a particular target CIM instance in the target  
 2054 namespace. Optionally, it retrieves a first set of instance paths.

```

2055 <instancePath>* OpenReferenceInstancePaths (
2056     [OUT] string EnumerationContext,
2057     [OUT] Boolean EndOfSequence,
2058     [IN] <instanceName> InstanceName,
2059     [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
2060     [IN,OPTIONAL,NULL] string Role = NULL,
2061     [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
2062     [IN,OPTIONAL,NULL] string FilterQuery = NULL,
2063     [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
2064     [IN,OPTIONAL] Boolean ContinueOnError = false,
2065     [IN,OPTIONAL] uint32 MaxObjectCount = 0
2066 )
  
```

2067 The OpenReferenceInstancePaths operation shall comply with the behavior defined in 5.4.2.24.1.

2068 The EnumerationContext output parameter is defined in 5.4.2.24.2.

2069 The EndOfSequence output parameter is defined in 5.4.2.24.2.

2070 The InstanceName input parameter specifies an instance name (model path) that identifies the target  
 2071 CIM instance with the referring association instances (respectively, their instance paths) to be  
 2072 enumerated. Unless restricted by any filter parameters of this operation, the enumeration set shall consist  
 2073 of the instance paths of all association instances that reference the target instance.

2074 The ResultClass input parameter, if not NULL, shall be a CIM class name. It acts as a filter on the  
 2075 enumerated set of instance paths by mandating that each enumerated instance path shall identify an  
 2076 instance of this class or one of its subclasses. The WBEM server shall not return an error if the  
 2077 ResultClass input parameter value is an invalid class name or if the class does not exist in the target  
 2078 namespace.

2079 The `Role` input parameter, if not NULL, shall be a property name. It acts as a filter on the enumerated set  
 2080 of instance paths by mandating that each enumerated instance path shall identify an instance that refers  
 2081 to the target instance through a property with a name that matches the value of this parameter. The  
 2082 WBEM server shall not return an error if the `Role` input parameter value is an invalid property name or if  
 2083 the property does not exist,

2084 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

2085 The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

2086 The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

2087 The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

2088 If `OpenReferenceInstancePaths` is successful, the return value shall be an array of `<instancePath>`  
 2089 items representing enumerated instance paths as defined in 5.4.2.24.2.

2090 The `PullInstancePaths` operation shall be used to pull instances for an enumeration session opened using  
 2091 `OpenReferenceInstancePaths`. If any other operation is used to pull instances, the WBEM server shall  
 2092 return failure with the status code `CIM_ERR_FAILED`.

2093 If `OpenReferenceInstancePaths` is unsuccessful, this operation shall return one of the following status  
 2094 codes, where the error returned is the first applicable error in the list, starting with the first element and  
 2095 working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- 2096 • `CIM_ERR_ACCESS_DENIED`
- 2097 • `CIM_ERR_SERVER_IS_SHUTTING_DOWN`
- 2098 • `CIM_ERR_NOT_SUPPORTED`
- 2099 • `CIM_ERR_INVALID_NAMESPACE`
- 2100 • `CIM_ERR_INVALID_OPERATION_TIMEOUT`
- 2101 • `CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED`
- 2102 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
 2103 incorrect parameters)
- 2104 • `CIM_ERR_NOT_FOUND` (The target instance was not found.)
- 2105 • `CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED`
- 2106 • `CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED` (The requested filter query language is  
 2107 not recognized.)
- 2108 • `CIM_ERR_INVALID_QUERY` (The filter query is not a valid query in the specified filter query  
 2109 language.)
- 2110 • `CIM_ERR_FAILED` (Some other unspecified error occurred.)

#### 2111 5.4.2.24.7 `OpenAssociatorInstances`

2112 The `OpenAssociatorInstances` operation establishes and opens an enumeration session of the instances  
 2113 associated with a particular source CIM instance in the target namespace. Optionally, it retrieves a first  
 2114 set of instances.

```
2115 <instanceWithPath>* OpenAssociatorInstances (
2116     [OUT] string EnumerationContext,
2117     [OUT] Boolean EndOfSequence,
2118     [IN] <instanceName> InstanceName,
```

```
2119     [IN,OPTIONAL,NULL] <className> AssocClass = NULL,  
2120     [IN,OPTIONAL,NULL] <className> ResultClass = NULL,  
2121     [IN,OPTIONAL,NULL] string Role = NULL,  
2122     [IN,OPTIONAL,NULL] string ResultRole = NULL,  
2123     [IN,OPTIONAL] boolean IncludeClassOrigin = false,  
2124     [IN,OPTIONAL,NULL] string PropertyList [] = NULL,  
2125     [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,  
2126     [IN,OPTIONAL,NULL] string FilterQuery = NULL,  
2127     [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,  
2128     [IN,OPTIONAL] Boolean ContinueOnError = false,  
2129     [IN,OPTIONAL] uint32 MaxObjectCount = 0  
2130 )
```

2131 The `OpenAssociatorInstances` operation shall comply with the behavior defined in 5.4.2.24.1.

2132 The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

2133 The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

2134 The `InstanceName` input parameter specifies an instance name (model path) that identifies the source  
2135 CIM instance with the associated instances to be enumerated. Unless restricted by any filter parameters  
2136 of this operation, the enumeration set shall consist of all instances associated with the source instance.

2137 The `AssocClass` input parameter, if not `NULL`, shall be a CIM association class name. It acts as a filter  
2138 on the enumerated set of instances by mandating that each enumerated instance shall be associated with  
2139 the source instance through an instance of this class or one of its subclasses. The WBEM server shall not  
2140 return an error if the `AssocClass` input parameter value is an invalid class name or if the class does not  
2141 exist in the target namespace.

2142 The `ResultClass` input parameter, if not `NULL`, must be a CIM class name. It acts as a filter on the  
2143 enumerated set of instances by mandating that each enumerated instance shall be an instance of this  
2144 class or one of its subclasses. The WBEM server shall not return an error if the `ResultClass` input  
2145 parameter value is an invalid class name or if the class does not exist in the target namespace.

2146 The `Role` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the enumerated set  
2147 of instances by mandating that each enumerated instance shall be associated with the source instance  
2148 through an association in which the source instance plays the specified role. That is, the name of the  
2149 property in the association class that refers to the source instance shall match the value of this  
2150 parameter. The WBEM server shall not return an error if the `Role` input parameter value is an invalid  
2151 property name or if the property does not exist.

2152 The `ResultRole` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the  
2153 enumerated set of instances by mandating that each enumerated instance shall be associated with the  
2154 source instance through an association in which the enumerated instance plays the specified role. That  
2155 is, the name of the property in the association class that refers to the enumerated instance shall match  
2156 the value of this parameter. The WBEM server shall not return an error if the `ResultRole` input  
2157 parameter value is an invalid property name or if the property does not exist.

2158 The `IncludeClassOrigin` input parameter is defined in 5.4.2.24.2.

2159 The `PropertyList` input parameter acts as a filter on the properties included in any enumerated CIM  
2160 instances. If `PropertyList` is not `NULL`, the members of the array define zero or more property names.  
2161 Each enumerated instance shall not include any properties missing from this list. If `PropertyList` is an  
2162 empty array, no properties are included in each enumerated instance. If `PropertyList` is `NULL`, all  
2163 properties are included in each enumerated instance, subject to the conditions expressed by the other  
2164 parameters. If `PropertyList` contains duplicate property names, the WBEM server shall ignore them

2165 but otherwise process the request normally. If `PropertyList` contains property names that are invalid  
 2166 for a target instance, the WBEM server shall ignore them for that instance but otherwise process the  
 2167 request normally. WBEM clients should not specify properties in `PropertyList` unless they specify a  
 2168 non-NULL value for the `ResultClass` parameter.

2169 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

2170 The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

2171 The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

2172 The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

2173 If `OpenAssociatorInstances` is successful, the return value shall be an array of `<instanceWithPath>`  
 2174 items representing enumerated instances as defined in 5.4.2.24.2.

2175 The `PullInstancesWithPath` operation shall be used to pull instances for an enumeration session opened  
 2176 using `OpenAssociatorInstances`. If any other operation is used to pull instances, the WBEM server shall  
 2177 return failure with the status code `CIM_ERR_FAILED`.

2178 If `OpenAssociatorInstances` is unsuccessful, this operation shall return one of the following status codes,  
 2179 where the error returned is the first applicable error in the list, starting with the first element and working  
 2180 down. Any additional operation-specific interpretation of the error is given in parentheses.

- 2181 • `CIM_ERR_ACCESS_DENIED`
- 2182 • `CIM_ERR_SERVER_IS_SHUTTING_DOWN`
- 2183 • `CIM_ERR_NOT_SUPPORTED`
- 2184 • `CIM_ERR_INVALID_NAMESPACE`
- 2185 • `CIM_ERR_INVALID_OPERATION_TIMEOUT`
- 2186 • `CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED`
- 2187 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
 2188 incorrect parameters)
- 2189 • `CIM_ERR_NOT_FOUND` (The source instance was not found.)
- 2190 • `CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED`
- 2191 • `CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED` (The requested filter query language is  
 2192 not recognized.)
- 2193 • `CIM_ERR_INVALID_QUERY` (The filter query is not a valid query in the specified filter query  
 2194 language.)
- 2195 • `CIM_ERR_FAILED` (Some other unspecified error occurred.)

#### 2196 5.4.2.24.8 `OpenAssociatorInstancePaths`

2197 The `OpenAssociatorInstancePaths` operation establishes and opens an enumeration session of the  
 2198 instance paths of the instances associated with a particular source CIM instance in the target namespace.  
 2199 Optionally, it retrieves a first set of instance paths.

```

2200 <instancePath>* OpenAssociatorInstancePaths (
2201     [OUT] string EnumerationContext,
2202     [OUT] Boolean EndOfSequence,
2203     [IN] <instanceName> InstanceName,
2204     [IN,OPTIONAL,NULL] <className> AssocClass= NULL,
```

```

2205     [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
2206     [IN,OPTIONAL,NULL] string Role = NULL,
2207     [IN,OPTIONAL,NULL] string ResultRole = NULL,
2208     [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
2209     [IN,OPTIONAL,NULL] string FilterQuery = NULL,
2210     [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
2211     [IN,OPTIONAL] Boolean ContinueOnError = false,
2212     [IN,OPTIONAL] uint32 MaxObjectCount = 0
2213 )

```

2214 This operation shall comply with the behavior defined in 5.4.2.24.1.

2215 The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

2216 The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

2217 The `InstanceName` input parameter specifies an instance name (model path) that identifies the source  
 2218 CIM instance with the associated instances (respectively, their instance paths) to be enumerated. Unless  
 2219 restricted by any filter parameters of this operation, the enumeration set shall consist of the instance  
 2220 paths of all instances associated with the source instance.

2221 The `AssocClass` input parameter, if not `NULL`, shall be a CIM association class name. It acts as a filter  
 2222 on the enumerated set of instance paths by mandating that each instance path identify an instance that  
 2223 shall be associated with the source instance through an instance of this class or one of its subclasses.  
 2224 The WBEM server shall not return an error if the `AssocClass` input parameter value is an invalid class  
 2225 name or if the class does not exist in the target namespace.

2226 The `ResultClass` input parameter, if not `NULL`, shall be a CIM class name. It acts as a filter on the  
 2227 enumerated set of instance paths by mandating that each instance path identify an instance that shall be  
 2228 an instance of this class or one of its subclasses. The WBEM server shall not return an error if the  
 2229 `ResultClass` input parameter value is an invalid class name or if the class does not exist in the target  
 2230 namespace.

2231 The `Role` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the enumerated set  
 2232 of instance paths by mandating that each instance path identify an instance that shall be associated with  
 2233 the source instance through an association in which the source instance plays the specified role. That is,  
 2234 the name of the property in the association class that refers to the source instance shall match the value  
 2235 of this parameter. The WBEM server shall not return an error if the `Role` input parameter value is an  
 2236 invalid property name or if the property does not exist.

2237 The `ResultRole` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the  
 2238 enumerated set of instance paths by mandating that each instance path identify an instance that shall be  
 2239 associated with the source instance through an association in which the instance identified by  
 2240 the enumerated instance path plays the specified role. That is, the name of the property in the association  
 2241 class that refers to the instance identified by the enumerated instance path shall match the value of this  
 2242 parameter. The WBEM server shall not return an error if the `ResultRole` input parameter value is an  
 2243 invalid property name or if the property does not exist.

2244 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

2245 The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

2246 The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

2247 The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

- 2248 If `OpenAssociatorInstancePaths` is successful, the return value shall be an array of `<instancePath>`  
 2249 items representing enumerated instance paths as defined in 5.4.2.24.2.
- 2250 The `PullInstancePaths` operation shall be used to pull instances for an enumeration session opened using  
 2251 `OpenAssociatorInstancePaths`. If any other operation is used to pull instances, the WBEM server shall  
 2252 return failure with the status code `CIM_ERR_FAILED`.
- 2253 If `OpenAssociatorInstancePaths` is unsuccessful, this operation shall return one of the following status  
 2254 codes, where the error returned is the first applicable error in the list, starting with the first element and  
 2255 working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.
- 2256 • `CIM_ERR_ACCESS_DENIED`
  - 2257 • `CIM_ERR_SERVER_IS_SHUTTING_DOWN`
  - 2258 • `CIM_ERR_NOT_SUPPORTED`
  - 2259 • `CIM_ERR_INVALID_NAMESPACE`
  - 2260 • `CIM_ERR_INVALID_OPERATION_TIMEOUT`
  - 2261 • `CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED`
  - 2262 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
 2263 incorrect parameters)
  - 2264 • `CIM_ERR_NOT_FOUND` (The source instance was not found.)
  - 2265 • `CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED`
  - 2266 • `CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED` (The requested filter query language is  
 2267 not recognized.)
  - 2268 • `CIM_ERR_INVALID_QUERY` (The filter query is not a valid query in the specified filter  
 2269 language.)
  - 2270 • `CIM_ERR_FAILED` (Some other unspecified error occurred.)

#### 2271 5.4.2.24.9 Common Parameters for the Pull Operations

- 2272 This clause defines commonly used parameters for the Pull operations. The description of the individual  
 2273 Pull operations references these parameters as appropriate. Note that not every Pull operation uses  
 2274 every one of these common parameters.
- 2275 • `EnumerationContext`
    - 2276 – This parameter is the enumeration context value representing the enumeration session to  
 2277 be used.
    - 2278 – The representation of an enumeration context value uses a string type. In version 1.3 of  
 2279 this document, enumeration context values were represented using the  
 2280 `ENUMERATIONCONTEXT` XML element. The representation was changed to using a  
 2281 string type in version 1.4 of this document, because it had turned out that all known  
 2282 implementations had implemented the enumeration context value using a string type.
    - 2283 – When the Pull operation is invoked, the enumeration session represented by the  
 2284 `EnumerationContext` input parameter shall be open. The first enumeration session shall  
 2285 use one of the Open operations with a type of enumerated object that matches the Pull  
 2286 operation. For the first Pull operation on an enumeration session, the value of the  
 2287 `EnumerationContext` input parameter shall be the enumeration context value returned  
 2288 by a successful Open operation. For subsequent Pull operations on that enumeration  
 2289 session, the value of the `EnumerationContext` input parameter shall be the value of the

- 2290 EnumerationContext output parameter returned by the previous Pull operation on the  
2291 same enumeration session.
- 2292 – After the Pull operation is completed, the enumeration session represented by the  
2293 EnumerationContext output parameter shall be open or closed.
- 2294 • EndOfSequence
    - 2295 – This output parameter indicates to the WBEM client whether the enumeration session is  
2296 exhausted. If EndOfSequence is true upon successful completion of a Pull operation, no  
2297 more instances or instance paths are available and the WBEM server shall close the  
2298 enumeration session, releasing any allocated resources related to the session. If  
2299 EndOfSequence is false, additional instances or instance paths may be available, and  
2300 the WBEM server shall not close the session.
  - 2301 • MaxObjectCount
    - 2302 – This input parameter defines the maximum number of instances or instance paths that may  
2303 be returned by this Pull operation. Any uint32 number is valid, including 0. The WBEM  
2304 server may deliver any number of instances or instance paths up to MaxObjectCount but  
2305 shall not deliver more than MaxObjectCount. The WBEM client may use a  
2306 MaxObjectCount value of 0 to restart the operation timeout for the enumeration session  
2307 when it does not need to not retrieve any instances or instance paths.
  - 2308 • Return Value (array of enumerated elements)
    - 2309 – The return value of a Pull operation upon successful completion is an array of enumerated  
2310 instances or instance paths with a number of entries from 0 up to a maximum defined by  
2311 MaxObjectCount. These entries meet the criteria defined in the Open operation that  
2312 established this enumeration session. Note that returning no entries in the array does not  
2313 imply that the enumeration session is exhausted. Only the EndOfSequence output  
2314 parameter indicates whether the enumeration session is exhausted.

#### 2315 5.4.2.24.10 PullInstancesWithPath

2316 The PullInstancesWithPath operation retrieves instances including their instance paths from an open  
2317 enumeration session represented by an enumeration context value:

```
2318 <instanceWithPath>* PullInstancesWithPath (
2319     [IN,OUT] string EnumerationContext,
2320     [OUT] Boolean EndOfSequence,
2321     [IN] uint32 MaxObjectCount
2322 )
```

2323 The PullInstancesWithPath operation shall comply with the behavior defined in 5.4.2.24.1.

2324 The EnumerationContext input/output parameter is defined in 5.4.2.24.9. The enumeration session  
2325 shall be established using one of the OpenEnumerateInstances, OpenReferenceInstances, or  
2326 OpenAssociatorInstances operations.

2327 The EndOfSequence output parameter is defined in 5.4.2.24.9.

2328 The MaxObjectCount input parameter is defined in 5.4.2.24.9.

2329 If PullInstancesWithPath is successful, the return value shall be an array of <instanceWithPath>  
2330 items representing enumerated instances including their instance paths as defined in 5.4.2.24.9.

2331 If PullInstancesWithPath is unsuccessful, this operation shall return one of the following status codes,  
2332 where the error returned is the first applicable error in the list, starting with the first element and working  
2333 down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- 2334 • CIM\_ERR\_ACCESS\_DENIED
- 2335 • CIM\_ERR\_SERVER\_IS\_SHUTTING\_DOWN
- 2336 • CIM\_ERR\_NOT\_SUPPORTED
- 2337 • CIM\_ERR\_INVALID\_NAMESPACE
- 2338 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise
- 2339 incorrect parameters)
- 2340 • CIM\_ERR\_INVALID\_ENUMERATION\_CONTEXT
- 2341 • CIM\_ERR\_PULL\_HAS\_BEEN\_ABANDONED
- 2342 • CIM\_ERR\_SERVER\_LIMITS\_EXCEEDED
- 2343 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 2344 5.4.2.24.11 PullInstancePaths

2345 The PullInstancePaths operation retrieves instance paths from an open enumeration session represented  
2346 by an enumeration context value:

```
2347     <instancePath>* PullInstancePaths (
2348         [IN,OUT] string EnumerationContext,
2349         [OUT] Boolean EndOfSequence,
2350         [IN] uint32 MaxObjectCount
2351     )
```

2352 The PullInstancePaths operation shall comply with the behavior defined in 5.4.2.24.1.

2353 The EnumerationContext input/output parameter is defined in 5.4.2.24.9. The enumeration session  
2354 shall have been established using one of the OpenEnumerateInstancePaths,  
2355 OpenReferenceInstancePaths, or OpenAssociatorInstancePaths operations.

2356 The EndOfSequence output parameter is defined in 5.4.2.24.9.

2357 The MaxObjectCount input parameter is defined in 5.4.2.24.9.

2358 If PullInstancePaths is successful, the return value shall be an array of <instancePath> items  
2359 representing enumerated instance paths as defined in 5.4.2.24.9.

2360 If PullInstancePaths is unsuccessful, this operation shall return one of the following status codes, where  
2361 the error returned is the first applicable error in the list, starting with the first element and working down.  
2362 Any additional operation-specific interpretation of the error is enclosed in parentheses.

- 2363 • CIM\_ERR\_ACCESS\_DENIED
- 2364 • CIM\_ERR\_SERVER\_IS\_SHUTTING\_DOWN
- 2365 • CIM\_ERR\_NOT\_SUPPORTED
- 2366 • CIM\_ERR\_INVALID\_NAMESPACE
- 2367 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise
- 2368 incorrect parameters)
- 2369 • CIM\_ERR\_INVALID\_ENUMERATION\_CONTEXT
- 2370 • CIM\_ERR\_SERVER\_LIMITS\_EXCEEDED
- 2371 • CIM\_ERR\_PULL\_HAS\_BEEN\_ABANDONED

- CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 2373 5.4.2.24.12 CloseEnumeration

2374 The CloseEnumeration operation closes an open enumeration session, performing an early termination of  
2375 an enumeration sequence:

```
2376 void CloseEnumeration (
2377     [IN] string EnumerationContext
2378 )
```

2379 The EnumerationContext parameter is the value representing the enumeration session to be closed.  
2380 The enumeration session shall be open and shall be established using one of the Open operations. This  
2381 implies that this operation is not to close an enumeration sequence already indicated by  
2382 EndOfSequence because the sequence has already been closed. The value of the  
2383 EnumerationContext parameter shall be the value of the EnumerationContext output parameter  
2384 returned by the previous Pull operation on the enumeration session to be closed.

2385 If CloseEnumeration is successful, the WBEM server shall close the enumeration session represented by  
2386 EnumerationContext, releasing any allocated resources. Any subsequent use of the  
2387 EnumerationContext value is unsuccessful.

2388 CloseEnumeration may be executed concurrently with a Pull operation or an EnumerationCount operation  
2389 on the same enumeration session. If a WBEM server receives a CloseEnumeration operation request  
2390 while it is processing a Pull operation on the same enumeration session, the WBEM server shall attempt  
2391 to abandon that Pull operation. If the Pull operation can be abandoned, it shall return a failure with the  
2392 status code [CIM\\_ERR\\_PULL\\_HAS\\_BEEN\\_ABANDONED](#) and the CloseEnumeration operation shall  
2393 return success. If the Pull operation cannot be abandoned, it shall proceed as if the CloseEnumeration  
2394 operation has not been issued, and the CloseEnumeration operation shall return a failure with the status  
2395 code [CIM\\_ERR\\_PULL\\_CANNOT\\_BE\\_ABANDONED](#).

2396 If CloseEnumeration is unsuccessful, this operation shall return one of the following status codes, where  
2397 the error returned is the first applicable error in the list, starting with the first element and working down.  
2398 Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_SERVER\_IS\_SHUTTING\_DOWN
- CIM\_ERR\_NOT\_SUPPORTED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM\_ERR\_INVALID\_ENUMERATION\_CONTEXT
- CIM\_ERR\_PULL\_CANNOT\_BE\_ABANDONED
- CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 2408 5.4.2.24.13 EnumerationCount

2409 The EnumerationCount operation provides an estimated count of the total number of objects in an open  
2410 enumeration session represented by an EnumerationContext:

```
2411 uint64 EnumerationCount (
2412     [IN] string EnumerationContext
2413 )
```

2414 The `EnumerationContext` parameter identifies the enumeration session for the `EnumerationCount`  
 2415 operation. It shall be established using any of the `Open` operations and shall be open at the time of the  
 2416 `CloseEnumeration` request. A conformant WBEM server may support this operation. A WBEM server that  
 2417 does not support this operation should respond with the [CIM\\_ERR\\_NOT\\_SUPPORTED](#) status.

2418 If `EnumerationCount` is successful, the operation returns an approximate count of the number of objects  
 2419 in the enumeration session. This is the number of items remaining to be sent with subsequent `Pull`  
 2420 operations. Thus, executing this operation immediately after the open may provide an approximate  
 2421 estimate of the total number of objects to be returned in the enumeration set. The returned count is only  
 2422 an estimate of the number of objects to be pulled in the enumeration sequence. This mechanism is  
 2423 intended to assist WBEM clients in determining the overall size of an enumeration set and the number of  
 2424 objects remaining in the enumeration session. It should not be used instead of the `EndOfSequence`  
 2425 parameter to determine the end of an enumeration sequence.

2426 If the WBEM server cannot or will not return an estimate of the number of objects to be returned for the  
 2427 enumeration context, it may return success and the NULL value.

2428 If `EnumerationCount` is unsuccessful, this operation shall return one of the following status codes, where  
 2429 the error returned is the first applicable error in the list, starting with the first element and working down.  
 2430 Any additional operation-specific interpretation of the error is enclosed in parentheses.

- 2431 • `CIM_ERR_ACCESS_DENIED`
- 2432 • `CIM_ERR_SERVER_IS_SHUTTING_DOWN`
- 2433 • `CIM_ERR_NOT_SUPPORTED`
- 2434 • `CIM_ERR_INVALID_NAMESPACE`
- 2435 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
 2436 incorrect parameters)
- 2437 • `CIM_ERR_INVALID_ENUMERATION_CONTEXT`
- 2438 • `CIM_ERR_SERVER_LIMITS_EXCEEDED`
- 2439 • `CIM_ERR_FAILED` (Some other unspecified error occurred.)

#### 2440 **5.4.2.24.14OpenQueryInstances**

2441 The `OpenQueryInstances` operation establishes and opens an enumeration session of the instances of a  
 2442 CIM class (including instances of its subclasses) in the target namespace. Optionally, it retrieves a first  
 2443 set of instances:

```

2444     <instance>* OpenQueryInstances (
2445         [IN] string FilterQuery,
2446         [IN] string FilterQueryLanguage,
2447         [IN,OPTIONAL] Boolean ReturnQueryResultClass = false,
2448         [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
2449         [IN,OPTIONAL] Boolean ContinueOnError = false,
2450         [IN,OPTIONAL] uint32 MaxObjectCount = 0,
2451         [OUT, OPTIONAL, NULL] <class> QueryResultClass,
2452         [OUT] string EnumerationContext,
2453         [OUT] Boolean EndOfSequence
2454     )
  
```

2455 The `OpenQueryInstances` shall comply with the behavior defined in 5.4.2.24.1.

2456 The `FilterQuery` and `FilterQueryLanguage` input parameters specify the set of enumerated  
 2457 instances.

- 2458 `FilterQueryLanguage` shall specify a query language and the value of `FilterQuery` shall be a valid  
2459 query in that query language. This document defines neither the query language nor the format of the  
2460 query. It is anticipated that query languages will be submitted to the DMTF as separate proposals. A  
2461 mechanism by which WBEM servers can declare the query languages they support for filtering in Pulled  
2462 enumerations (if any) is defined in 7.5.
- 2463 The `ReturnQueryResultClass` input parameter controls whether a class definition is returned in  
2464 `QueryResultClass`. If it is set to `false`, `QueryResultClass` shall be set to `NULL` on output. If it is  
2465 set to `true`, the value of the `QueryResultClass` on output shall be a class definition that defines the  
2466 properties (columns) of each row of the query result.
- 2467 The `OperationTimeout` input parameter is defined in 5.4.2.24.2.
- 2468 The `ContinueOnError` input parameter is defined in 5.4.2.24.2.
- 2469 The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.
- 2470 The `QueryResultClass` output parameter shall be set to `NULL` if the `ReturnQueryResultClass`  
2471 input parameter is set to `false`. Otherwise, it shall return a class definition where each property of the  
2472 class corresponds to one entry of the query select list. The class definition corresponds to one row of the  
2473 query result. The class name of this returned class shall be "CIM\_QueryResult." This class definition is  
2474 valid only in the context of this enumeration.
- 2475 The `EnumerationContext` output parameter is defined in 5.4.2.24.2.
- 2476 The `EndOfSequence` output parameter is defined in 5.4.2.24.2.
- 2477 If `OpenQueryInstances` is successful, the return value shall be an array of `<instance>` items  
2478 representing enumerated instances as defined in 5.4.2.24.2. Such instances are available only in the  
2479 context of the enumeration and do not return an instance path. The `PullInstancesWithPath` operation may  
2480 not be used to continue an enumeration started by the `OpenQueryInstances` operation.
- 2481 The `PullInstances` operation shall be used to pull instances for an enumeration session opened using `If`  
2482 `OpenQueryInstances`. If any other operation is used to pull instances, the WBEM server shall return  
2483 failure with the status code `CIM_ERR_FAILED`.
- 2484 If `OpenQueryInstances` is unsuccessful, this operation shall return one of the following status codes,  
2485 where the error returned is the first applicable error in the list, starting with the first element and working  
2486 down. Any additional operation-specific interpretation of the error is enclosed in parentheses.
- 2487 • `CIM_ERR_ACCESS_DENIED`
  - 2488 • `CIM_ERR_SERVER_IS_SHUTTING_DOWN`
  - 2489 • `CIM_ERR_NOT_SUPPORTED`
  - 2490 • `CIM_ERR_INVALID_NAMESPACE`
  - 2491 • `CIM_ERR_INVALID_OPERATION_TIMEOUT`
  - 2492 • `CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED`
  - 2493 • `CIM_ERR_INVALID_PARAMETER` (including missing, duplicate, unrecognized, or otherwise  
2494 incorrect parameters)
  - 2495 • `CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED` (The requested filter query language is  
2496 not recognized.)
  - 2497 • `CIM_ERR_INVALID_QUERY` (The filter query is not a valid query in the specified filter query  
2498 language.)

- 2499 • CIM\_ERR\_QUERY\_FEATURE\_NOT\_SUPPORTED (The query requires support for features  
2500 that are not supported.)
- 2501 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

#### 2502 5.4.2.24.15 PullInstances

2503 The PullInstances operation retrieves instances from an OpenQueryInstances session represented by an  
2504 enumeration context value:

```
2505 <instance>* PullInstances (
2506     [IN,OUT] string EnumerationContext,
2507     [OUT] Boolean EndOfSequence,
2508     [IN] uint32 MaxObjectCount
2509 )
```

2510 The PullInstances operation shall comply with the behavior defined in 5.4.2.24.1.

2511 The EnumerationContext input/output parameter is defined in 5.4.2.24.9. The enumeration session  
2512 shall be established using the OpenQueryInstances operation.

2513 The EndOfSequence output parameter is defined in 5.4.2.24.9.

2514 The MaxObjectCount input parameter is defined in 5.4.2.24.9.

2515 If PullInstances is successful, the return value shall be an array of <instance> items representing  
2516 enumerated instances as defined in 5.4.2.24.9.

2517 If PullInstances is unsuccessful, this operation shall return one of the following status codes, where the  
2518 error returned is the first applicable error in the list, starting with the first element and working down. Any  
2519 additional operation-specific interpretation of the error is enclosed in parentheses.

- 2520 • CIM\_ERR\_ACCESS\_DENIED
- 2521 • CIM\_ERR\_SERVER\_IS\_SHUTTING\_DOWN
- 2522 • CIM\_ERR\_NOT\_SUPPORTED
- 2523 • CIM\_ERR\_INVALID\_NAMESPACE
- 2524 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
2525 incorrect parameters)
- 2526 • CIM\_ERR\_INVALID\_ENUMERATION\_CONTEXT
- 2527 • CIM\_ERR\_SERVER\_LIMITS\_EXCEEDED
- 2528 • CIM\_ERR\_PULL\_HAS\_BEEN\_ABANDONED
- 2529 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

### 2530 5.4.3 Namespace Manipulation Using the CIM\_Namespace Class

2531 No intrinsic methods are defined specifically to manipulate namespaces. Namespaces shall be  
2532 manipulated using intrinsic methods on the CIM\_Namespace class.

#### 2533 5.4.3.1 Namespace Creation

2534 A namespace is created by calling the intrinsic method CreateInstance for the CIM\_Namespace class. A  
2535 value is specified for the new instance parameter that defines a valid instance of the CIM\_Namespace  
2536 class and that has a name property that is the desired name of the new namespace.

2537 The proposed definition shall be a correct namespace definition according to [DSP0004](#). Despite the  
2538 naming conventions used in the CIM specifications (use of / in namespaces such as root/CIMV2 and  
2539 root/CIMV2/test), there is no hierarchy implied among different namespaces. Each namespace is  
2540 independent of all others. The namespaces are to be considered flat, and there is no defined behavior for  
2541 navigating namespaces.

2542 In creating the new namespace, the WBEM server shall conform to the following rules:

- 2543 • The namespace defined by name property shall not already exist in the WBEM server.
- 2544 • The <LOCALNAMESPACEPATH> defined for the operation defines the namespace in which  
2545 the CIM\_Namespace instance associated with this new namespace is created.

2546 It is recommended that instances of CIM\_Namespace be created in root unless there is a specific reason  
2547 to define them in another namespace. The inclusion of a CIM\_Namespace instance within a namespace  
2548 other than root is allowed.

2549 In addition to creating instances of CIM\_Namespace, compliant implementations shall also create an  
2550 instance of the association class CIM\_NamespaceInManager defining the linking of the namespace  
2551 created to the current CIM\_ObjectManager.

2552 If CreateInstance is successful, the WBEM server creates the specified namespace. In addition, the  
2553 WBEM server shall return information about the namespace as an instance of the class CIM\_Namespace  
2554 and of returning instances of the association class CIM\_NamespaceInManager for each  
2555 CIM\_Namespace instance created.

#### 2556 **5.4.3.2 Namespace Deletion**

2557 If the WBEM server supports the CIM\_Namespace class, all valid namespaces shall be represented by  
2558 an instance of the CIM\_Namespace class. A namespace is deleted using the intrinsic method  
2559 DeleteInstance to delete the instance of the class CIM\_Namespace that represents the namespace. The  
2560 namespace to be deleted shall exist.

2561 If DeleteInstance is successful, the WBEM server shall remove the specified CIM\_Namespace instance.

2562 If DeleteInstance is unsuccessful, one of the status codes defined for the DeleteInstance operation shall  
2563 be returned. A WBEM server may return [CIM\\_ERR\\_FAILED](#) if a non-empty namespace cannot  
2564 successfully be deleted.

#### 2565 **5.4.3.3 Manipulation and Query of Namespace Information**

2566 The query of namespaces is provided through the following means:

- 2567 • Query of the CIM\_Namespace class on an individual namespace
- 2568 • Use of the CIM\_NamespaceInManager association to link the target CIM\_ObjectManager and  
2569 the instances of CIM\_Namespace representing all namespaces defined in the target  
2570 CIM\_ObjectManager

#### 2571 **5.4.3.4 Use of the \_\_Namespace Pseudo Class (DEPRECATED)**

2572 In previous versions of this document, namespaces were manipulated through the pseudo class  
2573 \_\_Namespace as follows:

2574 No intrinsic methods are specifically defined for manipulating CIM namespaces. However, modeling a  
2575 CIM namespace using class \_\_Namespace, together with the requirement that the root namespace be  
2576 supported by all WBEM servers, implies that all namespace operations can be supported.

2577 For example, all child namespaces of a particular namespace are enumerated by calling the intrinsic  
2578 method EnumerateInstanceNames against the parent namespace, specifying a value for the ClassName

2579 parameter of \_\_Namespace. A child namespace is created by calling the intrinsic method CreateInstance  
2580 against the parent namespace, specifying a value for the NewInstance parameter that defines a valid  
2581 instance of the class \_\_Namespace and that has a name property that is the desired name of the new  
2582 namespace.

2583 **DEPRECATION NOTE:** The use of the \_\_Namespace class is DEPRECATED. In its place, use the  
2584 CIM\_Namespace class.

#### 2585 5.4.4 Functional Profiles

2586 To establish conformance, this clause partitions the [intrinsic methods](#) into functional groups.

2587 Support for a particular group does *not* guarantee that all invocations of a method in that group will  
2588 succeed. Rather, the exclusion of a group is a declaration that any attempt to call a method in that group  
2589 always returns [CIM\\_ERR\\_NOT\\_SUPPORTED](#).

2590 Mechanisms by which a [WBEM server](#) may declare the functional groups that it supports are defined in  
2591 7.5.

2592 To limit the number of different profiles that a WBEM server may support, each functional group has a  
2593 dependency on another group (with the exception of the Basic Read functional group). If functional group  
2594 G<sub>1</sub> has a dependency on functional group G<sub>2</sub>, then a WBEM server that supports G<sub>1</sub> shall also support  
2595 G<sub>2</sub>.

2596 The dependency relation is transitive, so if G<sub>1</sub> depends on G<sub>2</sub>, and G<sub>2</sub> depends on G<sub>3</sub>, then G<sub>1</sub> depends  
2597 on G<sub>3</sub>. It is also anti-symmetric, so if G<sub>1</sub> depends on G<sub>2</sub>, then G<sub>2</sub> cannot depend on G<sub>1</sub>.

2598 Using these rules, Table 3 defines a rooted-directed tree of dependencies with the Basic Read  
2599 dependency representing the root node.

2600 For example, a WBEM server that supports the Schema Manipulation functional group shall also support  
2601 the Instance Manipulation, Basic Write, and Basic Read.

2602 A WBEM server shall support the Basic Read functional group.

2603

**Table 3 – Root-Directed Tree of Functional Profile Dependencies**

Functional Group	Dependency	Methods
Basic Read	none	<a href="#">GetClass</a> <a href="#">EnumerateClasses</a> <a href="#">EnumerateClassNames</a> <a href="#">GetInstance</a> <a href="#">EnumerateInstances</a> <a href="#">EnumerateInstanceNames</a> <a href="#">GetProperty (DEPRECATED)</a>
Pulled Read	Basic Read	<a href="#">OpenEnumerateInstances</a> <a href="#">OpenEnumerateInstancePaths</a> <a href="#">OpenReferenceInstances</a> <a href="#">OpenReferenceInstancePaths</a> <a href="#">OpenAssociatorInstances</a> <a href="#">OpenAssociatorInstancePaths</a> <a href="#">PullInstancesWithPath</a> <a href="#">PullInstancePaths</a> <a href="#">CloseEnumeration</a>
PulledReadCount	Pulled Read	<a href="#">EnumerationCount</a>
Pulled Query Execution	Pulled Read	<a href="#">OpenQueryInstances</a> <a href="#">PullInstances</a>
Basic Write	Basic Read	<a href="#">SetProperty (DEPRECATED)</a>
Schema Manipulation	Instance Manipulation	<a href="#">CreateClass</a> <a href="#">ModifyClass</a> <a href="#">DeleteClass</a>
Instance Manipulation	Basic Write	<a href="#">CreateInstance</a> <a href="#">ModifyInstance</a> <a href="#">DeleteInstance</a>
Association Traversal	Basic Read	<a href="#">Associators</a> <a href="#">AssociatorNames</a> <a href="#">References</a> <a href="#">ReferenceNames</a>
Query Execution	Basic Read	<a href="#">ExecQuery</a>
Qualifier Declaration	Schema Manipulation	<a href="#">GetQualifier</a> <a href="#">SetQualifier</a> <a href="#">DeleteQualifier</a> <a href="#">EnumerateQualifiers</a>

2604 **5.4.5 Extrinsic Method Invocation**

2605 Any [WBEM server](#) is assumed to support extrinsic methods, which are defined by the schema supported  
 2606 by the WBEM server. If a WBEM server does not support extrinsic method invocations, it shall return the  
 2607 error code [CIM\\_ERR\\_NOT\\_SUPPORTED](#) to any request to execute an extrinsic method (subject to the  
 2608 considerations described in the rest of this clause). This allows a [WBEM client](#) to determine that all  
 2609 attempts to execute extrinsic methods will fail.

2610 If the WBEM server cannot invoke extrinsic methods, it shall return one of the following status codes,  
2611 where the error returned is the first applicable error in the list, starting with the first element and working  
2612 down. Any additional specific interpretation of the error is enclosed in parentheses.

- 2613       • CIM\_ERR\_ACCESS\_DENIED
- 2614       • CIM\_ERR\_NOT\_SUPPORTED (The WBEM server does not support extrinsic method  
2615        invocations.)
- 2616       • CIM\_ERR\_INVALID\_NAMESPACE
- 2617       • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
2618        incorrect parameters)
- 2619       • CIM\_ERR\_NOT\_FOUND (The target CIM class or instance does not exist in the specified  
2620        namespace.)
- 2621       • CIM\_ERR\_METHOD\_NOT\_FOUND
- 2622       • CIM\_ERR\_METHOD\_NOT\_AVAILABLE (The WBEM server is unable to honor the invocation  
2623        request.)
- 2624       • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

## 2625 5.5 CIM Export Syntax and Semantics

2626 This clause focuses on export methods and their invocation, as well as on functional profiles.

### 2627 5.5.1 Export Method Invocations

2628 All CIM-XML export message requests defined for the CIM-to-HTTP mapping are invocations of one or  
2629 more export methods. Export methods do not operate against CIM namespaces.

2630 An export method call is represented in XML by the <EXPMETHODCALL> element, and the response to  
2631 that call is represented by the <EXPMETHODRESPONSE> element.

2632 An input parameter has an IN qualifier with value `true` in the method definition. An output parameter has  
2633 an OUT qualifier with value `true` in the method definition. A parameter may be both an input parameter  
2634 and an output parameter.

2635 The <EXPMETHODCALL> element names the method to be invoked and supplies any input parameters  
2636 to the export method call:

- 2637       • Each input parameter shall be named using the name assigned in the method definition.
- 2638       • Input parameters may be supplied in any order.
- 2639       • Each input parameter of the method, and no others, shall be present in the call unless it is  
2640        optional.

2641 The <EXPMETHODRESPONSE> element defines either an <ERROR> or a (possibly optional) return  
2642 value and output parameters, which are decorated with the OUT qualifier in the method definition. In the  
2643 latter case, the following rules apply:

- 2644       • Each output parameter shall be named using the name assigned in the method definition.
- 2645       • Output parameters may be supplied in any order.
- 2646       • Each output parameter of the method, and no others, shall be present in the response, unless it  
2647        is optional.

2648 The method invocation process may be thought of as a two-part process:

- 2649 • Binding the input parameter values specified as child elements of the <EXPMETHODCALL>  
2650 element to the input parameters of the method.
- 2651 • Attempting to execute the method using the bound input parameters, with one of the following  
2652 results:
  - 2653 – If the attempt to call the method is successful, the return value and output parameters are  
2654 bound to the child elements of the <EXPMETHODRESPONSE> element.
  - 2655 – If the attempt to call the method is unsuccessful, an error code and (optional) human-  
2656 readable description of that code is bound to the <EXPMETHODRESPONSE> element.

### 2657 5.5.1.1 Simple Export

2658 A simple export requires the invocation of a single export method. A simple export request is represented  
2659 by a <SIMPLEEXPREQ> element, and a simple export response is represented by a <SIMPLEEXPRSP>  
2660 element.

2661 A <SIMPLEEXPREQ> shall contain a <EXPMETHODCALL> element.

### 2662 5.5.1.2 Multiple Export

2663 A multiple export requires the invocation of more than one export method. A multiple export request is  
2664 represented by a <MULTIEXPREQ> element, and a multiple export response is represented by a  
2665 <MULTIEXPRSP> element.

2666 A <MULTIEXPREQ> (or its respective <MULTIEXPRSP>) element is a sequence of two or more  
2667 <SIMPLEEXPREQ> (or its respective <SIMPLEEXPRSP>) elements.

2668 A <MULTIEXPRSP> element shall contain a <SIMPLEEXPRSP> element for every <SIMPLEEXPREQ>  
2669 element in the corresponding multiple export response. These <SIMPLEEXPRSP> elements shall be in  
2670 the same order as their <SIMPLEEXPREQ> counterparts. The first <SIMPLEEXPRSP> in the response  
2671 corresponds to the first <SIMPLEEXPREQ> in the request, and so forth.

2672 Multiple exports conveniently batch the delivery of multiple export method invocations into a single HTTP  
2673 message, reducing the number of roundtrips between a WBEM client and a WBEM listener and allowing  
2674 the WBEM listener to make certain internal optimizations. Note that multiple exports do not confer any  
2675 transactional capabilities in processing the request. For example, the WBEM listener does not have to  
2676 guarantee that the constituent export method calls either all failed or all succeeded. The WBEM listener  
2677 must only make a "best effort" to process the operation. However, WBEM listeners shall finish processing  
2678 each method invocation in a batched message before executing the next method invocation in the batch.  
2679 Clients shall recognize that the order of method calls within a batched message is significant.

2680 Not all WBEM listeners support multiple exports. If a WBEM listener does not support multiple exports, it  
2681 shall return the status code CIM\_ERR\_NOT\_SUPPORTED.

### 2682 5.5.1.3 Status Codes

2683 This clause defines the status codes and detailed error information that a conforming WBEM listener may  
2684 return.

2685 The value of an <ERROR> child element within a <EXPMETHODRESPONSE> element includes the  
2686 following parts:

- 2687 • mandatory status code
- 2688 • optional human-readable description of the status code
- 2689 • zero or more CIM\_Error instances

2690 The symbolic names defined in Table 4 do not appear on the wire. They are used here solely for  
 2691 convenient reference to an error in other parts of this document. Not all methods are expected to return  
 2692 all these status codes.

2693 In addition to returning a status code, a conforming WBEM listener may return zero or more  
 2694 <INSTANCE> child elements as part of an <ERROR> element. Each <INSTANCE> child element shall  
 2695 be an instance of CIM\_Error, and the value of CIMStatusCode shall comply with the definition of expected  
 2696 error codes for the CIM-XML export request. A WBEM client may ignore any <INSTANCE> child  
 2697 elements.

2698 **Table 4 – Symbolic Names for Referencing Error Codes**

Symbolic Name	Code	Definition
CIM_ERR_FAILED	1	A general error occurred that is not covered by a more specific error code.
CIM_ERR_ACCESS_DENIED	2	Access was not available to the client.
CIM_ERR_NOT_SUPPORTED	7	The requested operation is not supported.
CIM_ERR_TYPE_MISMATCH	13	The value supplied is incompatible with the type.

2699 **5.5.2 Export Methods**

2700 This clause describes the methods that can be defined within a CIM-XML export message. These  
 2701 methods operate only on an external data representation of a CIM entity, namespace, or element.  
 2702 Specifically, export methods do not operate on CIM namespaces or elements. The export method defined  
 2703 in this document is Export an Indication.

2704 The notation used in the following subclauses to define the signatures of the export methods is a pseudo-  
 2705 MOF notation that extends the standard MOF BNF ([DSP0004](#)) for describing CIM export methods with a  
 2706 number of pseudo parameter types. The pseudo parameter types are enclosed in angle brackets (< >).

2707 This notation allows parameters to be decorated with pseudo-qualifiers (IN, OPTIONAL, and NULL) to  
 2708 define their invocation semantics. Note that these qualifiers are for description purposes only within the  
 2709 scope of this document. In particular, a WBEM client shall not specify them in export method invocations.

2710 This notation uses the IN qualifier for input parameters.

2711 A WBEM client may omit an optional parameter if the required value is the specified default by not  
 2712 specifying an <EXPPARAMVALUE> element for the parameter. It shall not omit a parameter that is not  
 2713 optional.

2714 The NULL qualifier indicates parameters with values that may be specified as NULL in an export method  
 2715 call. A NULL (unassigned) value for a parameter is specified by an <EXPPARAMVALUE> element with  
 2716 no child element. The WBEM client shall specify a value for parameters without the NULL qualifier by  
 2717 including a suitable child element for the <EXPPARAMVALUE> element.

2718 All parameters shall be uniquely named and shall correspond to a valid parameter name for that method  
 2719 as described by this document. The order of the parameters is not significant.

2720 The non-NULL values of export method parameters or return values that are modeled as standard CIM  
 2721 types (such as string and Boolean, or arrays thereof) are represented as follows:

- 2722 • Simple values shall be represented by the <VALUE> child element in an <EXPPARAMVALUE>  
 2723 element (for export method parameters) or in an <IRETURNVALUE> element (for export  
 2724 method return values).

2725           •    Array values shall be represented by the <VALUE.ARRAY> child element in an  
2726                    <EXPPARAMVALUE> element (for export method parameters) or in an <IRETURNVALUE>  
2727                    element (for export method return values).

2728    Table 5 shows how each pseudo-type used by the export methods shall be mapped to an XML element  
2729    described in [DSP0201](#) in the context of both a parameter value (child element of <EXPPARAMVALUE>)  
2730    and a return value (child element of <IRETURNVALUE>).  
2731

2732

Table 5 – Mapping of Export Method Pseudo-Types to XML Elements

Type	XML Element
<object>	(VALUE.OBJECT VALUE.OBJECTWITHLOCALPATH VALUE.OBJECTWITHPATH)
<class>	CLASS
<instance>	INSTANCE
<className>	CLASSNAME
<namedInstance>	VALUE.NAMEDINSTANCE
<instanceName>	INSTANCENAME
<objectWithPath>	VALUE.OBJECTWITHPATH
<objectName>	(CLASSNAME INSTANCENAME)
<propertyValue>	(VALUE VALUE.ARRAY VALUE.REFERENCE)
<qualifierDecl>	QUALIFIER.DECLARATION

### 2733 5.5.2.1 ExportIndication

2734 The ExportIndication operation exports a single CIM indication to the destination WBEM listener:

```
2735 void ExportIndication (
2736     [IN] <instance> NewIndication
2737 )
```

2738 The `NewIndication` input parameter defines the indication to be exported. The proposed definition  
2739 should be a correct instance definition for the underlying CIM indication class according to the [CIM](#)  
2740 [specification](#).

2741 If ExportIndication is unsuccessful, this method shall return one of the following status codes, where the  
2742 error returned is the first applicable error in the list, starting with the first element and working down. Any  
2743 additional method-specific interpretation of the error is enclosed in parentheses.

- 2744 • CIM\_ERR\_ACCESS\_DENIED
- 2745 • CIM\_ERR\_NOT\_SUPPORTED
- 2746 • CIM\_ERR\_INVALID\_PARAMETER (including missing, duplicate, unrecognized, or otherwise  
2747 incorrect parameters)
- 2748 • CIM\_ERR\_INVALID\_CLASS (The CIM class of which this is to be a new instance does not  
2749 exist.)  
2750 **DEPRECATED:** The use of CIM\_ERR\_INVALID\_CLASS has been deprecated in version 1.4 of  
2751 this document because a WBEM listener has no notion about existing classes. Listeners should  
2752 not use this status code anymore, and WBEM servers receiving this status code should treat it  
2753 like CIM\_ERR\_FAILED.
- 2754 • CIM\_ERR\_FAILED (Some other unspecified error occurred.)

### 2755 5.5.3 Functional Profiles

2756 This clause partitions the export methods into functional groups to establish conformance. See Table 6.

2757 Support for a particular group does not guarantee that all invocations of an export method in that group  
2758 will succeed. Rather, the exclusion of a group is a declaration that any attempt to call an export method in  
2759 that group always returns CIM\_ERR\_NOT\_SUPPORTED.

2760 The dependency relation is transitive, so if group  $G_1$  depends on  $G_2$ , and  $G_2$  depends on  $G_3$ , then  $G_1$   
 2761 depends on  $G_3$ . It is also anti-symmetric, so if  $G_1$  depends on  $G_2$ , then  $G_2$  cannot depend on  $G_1$ .

2762 **Table 6 – Functional Groups of Export Methods**

Functional Group	Dependency	Method
Indication	None	ExportIndication

## 2763 6 Encapsulation of CIM-XML Messages

2764 This clause describes how to use CIM-XML messages in HTTP. CIM-XML message requests may be  
 2765 used with or without the [HTTP Extension Framework](#).

2766 Although CIM-XML messages can be used in combination with a variety of HTTP request methods, this  
 2767 document defines CIM-XML messages only within HTTP POST requests. (M-POST may be used in place  
 2768 of POST. For details on how to use CIM-XML messages with the HTTP Extension Framework, see 6.2.)

2769 All CIM-XML message responses are carried in the corresponding HTTP response. In the remaining  
 2770 discussion, the following terms are used as convenient shorthand for the definitions provided here:

- 2771 • *CIM-XML operation request.* An HTTP POST request message with an XML entity body that  
 2772 defines an [operation request message](#).
- 2773 • *CIM-XML operation response.* An HTTP response message, issued in response to a CIM-XML  
 2774 operation request, with an entity body that defines an [operation response message](#).
- 2775 • *CIM-XML export request.* An HTTP POST request message with an XML entity body that  
 2776 defines an [export request message](#).
- 2777 • *CIM-XML export response.* An HTTP response message, issued in response to a CIM-XML  
 2778 export message request, with an entity body that defines an [export response message](#).
- 2779 • *CIM-XML message request.* An HTTP POST request message with an XML entity body that  
 2780 defines either an [operation request message](#) or an [export request message](#).
- 2781 • *CIM-XML message response.* An HTTP response message, issued in response to a CIM-XML  
 2782 message request, with an entity body that defines either an [operation response message](#) or an  
 2783 [export response message](#).

2784 Note that an HTTP response to a CIM request is not always a CIM response. For example, a "505 HTTP  
 2785 Version Not Supported" response is not a CIM response.

### 2786 6.1 WBEM clients, WBEM servers, and WBEM listeners

2787 A *CIM product* is any product that can supply and/or consume management information using the CIM  
 2788 schema. In particular, WBEM clients, WBEM servers, and WBEM listeners are examples of CIM products:

- 2789 • A *WBEM client* issues [CIM-XML operation requests](#) and receives and processes [CIM-XML](#)  
 2790 [operation responses](#).
- 2791 • A *WBEM server* receives and processes [CIM-XML operation requests](#) and issues [CIM-XML](#)  
 2792 [operation responses](#). A WBEM server also issues [CIM-XML export requests](#) and receives and  
 2793 processes [CIM-XML export responses](#).
- 2794 • A *WBEM listener* is a server that receives and processes [CIM-XML export requests](#) and issues  
 2795 [CIM-XML export responses](#).

2796 Throughout this document, the terms WBEM client, WBEM server, WBEM listener, and CIM product are  
 2797 used as convenient shorthand to refer to the subset of CIM products that conform to this document.

2798 Note that "WBEM client" (server, listener) was used for the term "WBEM client" (server, listener) before  
2799 version 1.4 of this document.

## 2800 6.2 Use of M-POST

2801 A [WBEM client](#) attempting to invoke a CIM-XML message using the HTTP Extension Framework method  
2802 "M-POST" shall follow these steps:

- 2803 • If the M-POST invocation fails with an HTTP status of "501 Not Implemented" or "510 Not  
2804 Extended," the client should retry the request using the HTTP method "POST" with the  
2805 appropriate modifications (described in 6.2.2).
- 2806 • If the M-POST invocation fails with an HTTP status of "405 Method Not Allowed," the client  
2807 should fail the request.
- 2808 • For all other status codes, the client shall act in accordance with standard HTTP (see 7.1).

2809 This extended invocation mechanism gives Internet proxies and firewalls greater filtering control and  
2810 administrative flexibility over CIM-XML message invocations.

2811 If a client receives a 501 or 510 status in response to an M-POST request, in subsequent invocations to  
2812 the same HTTP server, the client may omit the attempt at M-POST invocations for a suitable period. This  
2813 omission avoids the need for an extra round trip on each and every method invocation. The details of the  
2814 caching strategy employed by the client are outside the scope of this document.

### 2815 6.2.1 Use of the Ext Header

2816 If a [WBEM server](#) or [WBEM listener](#) receives a valid M-POST request and has fulfilled all mandatory  
2817 extension header declarations in the request, it shall include in the response the "Ext" header defined by  
2818 [RFC2774](#). This included header shall be protected by the appropriate [Cache-Control](#) directive.

### 2819 6.2.2 Naming of Extension Headers

2820 In M-POST request messages (and their responses), CIM extension headers shall be declared using the  
2821 name space prefix allotted by the "Man" extension header (in accordance with [RFC2774](#)) that refers to  
2822 the name space "http://www.dmtf.org/cim/mapping/http/v1.0". The full format of the "Man" header  
2823 declaration for this document is:

```
2824 Man           = "Man" ":" "http://www.dmtf.org/cim/mapping/http/v1.0"
2825               ";" "ns" "=" header-prefix
2826
2827 header-prefix = 2*DIGIT
```

2828 This header-prefix should be generated at random on a per-HTTP message basis, and should not  
2829 necessarily be a specific number.

2830 In accordance with [RFC2774](#), all POST request messages (and their responses) shall not include such a  
2831 mandatory extension declaration. In POST request messages (and their responses), name space  
2832 prefixes shall not be used.

2833 EXAMPLE 1:

2834 Using M-POST:

```
2835 M-POST /cimom HTTP/1.1
2836 Man: http://www.dmtf.org./cim/mapping/http/v1.0 ; ns=23
2837 23-CIMOperation: MethodCall
2838 ...
```

2839 EXAMPLE 2:

2840 Using POST:  
2841 POST /cimom HTTP/1.1  
2842 CIMOperation: MethodCall  
2843 ...

### 2844 6.3 Extension Headers Defined for CIM-XML Message Requests and Responses

2845 A CIM-XML message contains exactly one CIM-XML operation request, CIM-XML operation response,  
2846 CIM-XML export request, or CIM-XML export response. This clause describes the extension headers to  
2847 specify CIM-XML message semantics in the HTTP header of a POST message.

2848 Any [CIM-XML operation request](#) or [CIM-XML operation response](#) shall, and only CIM-XML operation  
2849 requests and responses may, include the following CIM extension header:

- 2850 • [CIMOperation](#)

2851 Any [CIM-XML operation request](#) shall, and only CIM-XML operation requests may, include one and only  
2852 one of the following CIM extension header sets:

- 2853 • [CIMMethod](#) and [CIMObject](#), or
- 2854 • [CIMBatch](#)

2855 Any CIM-XML export request or CIM-XML export response shall, and only CIM-XML export requests and  
2856 responses may, include the following CIM extension header:

- 2857 • [CIMExport](#)

2858 Any CIM-XML export request shall, and only CIM-XML export requests may, include one and only one of  
2859 the following CIM extension headers:

- 2860 • [CIMExportMethod](#)
- 2861 • [CIMExportBatch](#)

2862 An HTTP response with an error status code to a CIM-XML message request may include the following  
2863 CIM extension header:

- 2864 • [CIMError](#)

2865 All CIM-XML messages may include the following CIM extension header:

- 2866 • [CIMProtocolVersion](#)

#### 2867 6.3.1 Encoding of CIM Element Names within HTTP Headers and Trailers

2868 CIM element (class, property, qualifier, method, or method parameter) names are natively Unicode, and  
2869 may use UCS-2 characters unsuitable for inclusion within an HTTP message header or trailer. To encode  
2870 CIM element names represented in Unicode to values within HTTP headers or trailers, the following two-  
2871 step mapping process shall be used:

- 2872 • Encode the full Unicode CIM element name using [UTF-8](#).
- 2873 • Using the ""%" HEX HEX" convention, apply the standard URI [[RFC2396](#), section 2] escaping  
2874 mechanism to the resulting string to escape any characters that are unsafe within an HTTP  
2875 header or trailer.

2876 In this document, the token CIMIdentifier represents a CIM element name to which this transformation  
2877 has been applied.

2878 One characteristic of this mapping is that CIM elements named with an ASCII representation appear in  
2879 ASCII in the resulting URL.

2880 EXAMPLES:

- 2881 • CIM\_LogicalElement is unchanged under this transformation.
- 2882 • The class named using the UCS-2 sequence representing the Hangul characters for the Korean  
2883 word "hangugo" (D55C, AD6D, C5B4) becomes

2884 `%ED%95%9C%EA%B5%AD%EC%96%B4=10`

2885 after UTF-8 transformation and escaping all characters with their % HEX HEX equivalent.

### 2886 6.3.2 Encoding of CIM Object Paths within HTTP Headers and Trailers

2887 This clause describes the mapping that shall be applied to represent CIM object paths, as described  
2888 within an [Operation Request Message](#) using the <LOCALNAMESPACEPATH>, <LOCALCLASSPATH>,  
2889 or <LOCALINSTANCEPATH> elements, in a format that is safe for representation within an HTTP header  
2890 or trailer.

2891 If the element to be transformed is a <LOCALNAMESPACEPATH>, the algorithm is as follows:

- 2892 • For the first <NAMESPACE> child element, output the textual content of that element.
- 2893 • For each subsequent <NAMESPACE> child element, output the forward slash character (/)  
2894 followed by the textual content of that <NAMESPACE> element.

2895 If the element to be transformed is a <LOCALCLASSPATH>, the algorithm is as follows:

- 2896 • Transform the <LOCALNAMESPACEPATH> child element using the rules previously  
2897 described, and output a colon character (:).
- 2898 • Output the value of the NAME attribute of the <CLASSNAME> child element.

2899 If the element to be transformed is a <LOCALINSTANCEPATH>, the algorithm is as follows:

- 2900 • Transform the <LOCALNAMESPACEPATH> child element using the rules previously  
2901 described, and output a colon character (:).
- 2902 • Output the value of the CLASSNAME attribute of the <INSTANCENAME> child element.
- 2903 • If there is at least one <KEYBINDING> child element under the <INSTANCENAME> child  
2904 element, then for each such child element:
  - 2905 – Output a period character (.) if this is the first <KEYBINDING> child element; otherwise,  
2906 output a comma character (,).
  - 2907 – Output the value of the NAME attribute, followed by an equal character (=).
  - 2908 – If there is a <KEYVALUE> child element, output the textual element content of that  
2909 element, subject to the following transformation:
    - 2910 • If the VALUETYPE attribute is numeric or Boolean, the output is identical to the  
2911 content of the element.
    - 2912 • If the VALUETYPE attribute is a string, the output is obtained by enclosing the content  
2913 of the element in double quote (") characters and escaping any double quote  
2914 characters or backslash character within the value with a preceding backslash (\)  
2915 character.
  - 2916 – If there is a <VALUE.REFERENCE> child element
    - 2917 • Output a double quote character (").

- 2918                   • Apply the process recursively to the <CLASSPATH> or <INSTANCEPATH> child  
2919                   element of the <VALUE.REFERENCE> element, escaping any double quote or  
2920                   backslash character thereby generated with a preceding backslash (\) character.
- 2921                   • Output a closing double quote character (").
- 2922                   • If there is no <KEYBINDING> child element but there is a <KEYVALUE> or  
2923                   <VALUE.REFERENCE> child element under the <INSTANCENAME> child element, then:
- 2924                   – Output an equal character (=).
- 2925                   – Output the transformed value of the <KEYVALUE> or <VALUE.REFERENCE> using the  
2926                   previously-described rules.
- 2927                   • If there are no <KEYBINDING> child elements or no <KEYVALUE> or <VALUE.REFERENCE>  
2928                   child element, then indicate a singleton instance by outputting the string "=@" under the  
2929                   <INSTANCENAME> child element.

2930 Finally, after applying these rules to the <LOCALNAMESPACEPATH>, <LOCALCLASSPATH>, or  
2931 <LOCALINSTANCEPATH> element, transform the entire output string into URI-safe format in the  
2932 following two-step procedure:

- 2933                   • Encode the string using UTF-8 [\[RFC2279\]](#) if it is not already in this format.
- 2934                   • Using the "%" HEX HEX" convention, apply the standard URI [\[RFC2396\]](#), section 2] escaping  
2935                   mechanism to the resulting string to escape any characters that are unsafe within an HTTP  
2936                   header or trailer.

2937 In this document, the token CIMObjectPath represents a <LOCALNAMESPACEPATH>,  
2938 <LOCALCLASSPATH>, or <LOCALINSTANCEPATH> element to which the preceding transformation  
2939 has been applied.

### 2940 6.3.3 CIMOperation

2941 The CIMOperation header shall be present in all [CIM-XML operation request](#) and [CIM-XML operation](#)  
2942 [response](#) messages. It identifies the HTTP message as carrying a CIM-XML operation request or  
2943 response.

2944                   CIMOperation = "CIMOperation" ":" ("MethodCall" | "MethodResponse")

2945 A [WBEM client](#) shall include this header, with the value "MethodCall," in all CIM-XML operation requests  
2946 that it issues. A [WBEM server](#) shall include this header in all CIM-XML operation responses that it issues,  
2947 with the value "MethodResponse".

2948 If a WBEM server receives a CIM-XML operation request with this header, but with a missing value or a  
2949 value that is not "MethodCall," then it shall fail the request with status "400 Bad Request". The WBEM  
2950 server shall include a [CIMError](#) header in the response with a value of `unsupported-operation`.

2951 If a WBEM server receives a CIM-XML operation request without this header, it shall not process it as a  
2952 CIM-XML operation request. The status code returned by the WBEM server in response to such a  
2953 request is outside the scope of this document.

2954 If a WBEM client receives a response to a CIM-XML operation request without this header (or if this  
2955 header has a value that is not "MethodResponse"), it should discard the response and take appropriate  
2956 measures to publicize that it has received an incorrect response. The details as to how this is done are  
2957 outside the scope of this document.

2958 The CIMOperation header affords a simple mechanism by which firewall or proxy administrators can  
2959 make global administrative decisions on all CIM operations.

2960 **6.3.4 CIMExport**

2961 The CIMExport header shall be present in all CIM-XML export request and response messages. It  
 2962 identifies the HTTP message as carrying a CIM export method request or response.

2963 `CIMExport = "CIMExport" ":" ("MethodRequest" | "MethodResponse")`

2964 A WBEM client shall include this header with the value "MethodRequest" in all CIM-XML export requests  
 2965 that it issues. A WBEM listener shall include this header in all CIM-XML export responses that it issues,  
 2966 with the value "MethodResponse".

2967 If a WBEM listener receives a CIM-XML export request with this header, but with a missing value or a  
 2968 value that is not "MethodRequest", then it shall fail the request with status "400 Bad Request". The  
 2969 WBEM listener shall include a CIMError header in the response with a value of unsupported-operation.

2970 If a WBEM listener receives a CIM-XML export request without this header, it shall not process it. The  
 2971 status code returned by the WBEM listener in response to such a request is outside of the scope of this  
 2972 document.

2973 If a WBEM client receives a response to a CIM-XML export request without this header (or if this header  
 2974 has a value that is not "MethodResponse"), it should discard the response and take appropriate  
 2975 measures to publicize that it has received an incorrect response. The details as to how this is done are  
 2976 outside the scope of this document.

2977 The CIMExport header affords a simple mechanism by which firewall or proxy administrators can make  
 2978 global administrative decisions on all CIM exports.

2979 **6.3.5 CIMProtocolVersion**

2980 The CIMProtocolVersion header may be present in any CIM-XML message. The header identifies the  
 2981 version of the CIM operations over the HTTP specification in use by the sending entity.

2982 `CIMProtocolVersion = "CIMProtocolVersion" ":" 1*DIGIT "." 1*DIGIT`

2983 If the header is omitted, then a value of 1.0 must be assumed.

2984 The major and minor revision numbers must be treated as independent integers.

2985 The CIMProtocolVersion  $x_1.y_1$  is less than CIMProtocolVersion  $x_2.y_2$  if and only if one of the following  
 2986 statements is true:

- 2987
- $x_1$  is less than  $x_2$
  - 2988 •  $x_1$  equals  $x_2$ , and  $y_1$  is less than  $y_2$

2989 The CIMProtocolVersion  $x_1.y_1$  is greater than CIMProtocolVersion  $x_2.y_2$  if and only if one of the following  
 2990 statements is true:

- 2991
- $x_1$  is greater than  $x_2$ ,
  - 2992 •  $x_1$  equals  $x_2$ , and  $y_1$  is greater than  $y_2$

2993 A CIMProtocolVersion  $x_1.y_1$  is within tolerance of CIMProtocolVersion  $x_2.y_2$  if:

- 2994
- $x_1$  equals  $x_2$ , and
  - 2995 •  $y_1$  is less than or equal to  $y_2$

2996 If the CIMProtocolVersion of the CIM-XML message received is within tolerance of the  
 2997 CIMProtocolVersion supported for a [WBEM server](#) or [WBEM listener](#) implementation, the receiving  
 2998 implementation shall accept that CIM-XML message. Equivalent CIMProtocolVersion values between  
 2999 [WBEM server](#) or [WBEM listener](#) and the [WBEM client](#) shall be accepted. The [WBEM server](#) or [WBEM](#)

3000 [listener](#) implementation may reject a CIM-XML message in all other cases. For information about how  
3001 CIM-XML messages are rejected, see 7.3.

3002 Beyond tolerance considerations, the implementation should reject the received CIM-XML message *only*  
3003 if the design as defined by the CIMProtocolVersion of the receiving implementation has changed in the  
3004 declaration of the API, method parameters, or behavior since the design defined by the  
3005 CIMProtocolVersion of the received CIM-XML message.

### 3006 6.3.6 CIMMethod

3007 The CIMMethod header shall be present in any [CIM-XML operation request](#) message that contains a  
3008 [Simple Operation Request](#).

3009 It shall not be present in any [CIM-XML operation response](#) message nor in any [CIM-XML operation](#)  
3010 [request](#) message unless it is a simple operation request. It shall not be present in any CIM-XML export  
3011 request or response message.

3012 The header identifies the name of the CIM method to be invoked, encoded in an [HTTP-safe](#)  
3013 [representation](#). Firewalls and proxies may use this header to carry out routing and forwarding decisions  
3014 based on the CIM method to be invoked.

3015 The name of the CIM method within a simple operation request is the value of the NAME attribute of the  
3016 <METHODCALL> or <IMETHODCALL> element.

```
3017     CIMMethod = "CIMMethod" ":" MethodName
```

3018

```
3019     MethodName = CIMIdentifier
```

3020 If a [WBEM server](#) receives a CIM-XML operation request for which any one of the following statements is  
3021 true, then it shall fail the request and return a status of "400 Bad Request". Also, it shall include a  
3022 [CIMError](#) header in the response with a value of `header-mismatch`, subject to the considerations  
3023 specified in 7.3:

- 3024 • The CIMMethod header is present, but it has an invalid value.
- 3025 • The CIMMethod header is not present, but the operation request message is a [Simple](#)  
3026 [Operation Request](#).
- 3027 • The CIMMethod header is present, but the operation request message is not a simple operation  
3028 request.
- 3029 • The CIMMethod header is present and the operation request message is a simple operation  
3030 request, but the CIMIdentifier value (when unencoded) does not match the unique method  
3031 name within the simple operation request.

3032 Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy was  
3033 not acting on misleading information when it decided to forward the request based on the content of the  
3034 CIMMethod header. Additional securing of HTTP messages against modification in transit (such as the  
3035 encryption of the payload or appending of a digital signature thereto) would be required to provide a  
3036 higher degree of integrity.

### 3037 6.3.7 CIMObject

3038 The CIMObject header shall be present in any [CIM-XML operation request](#) message that contains a  
3039 [Simple Operation Request](#).

3040 It shall not be present in any [CIM-XML operation response](#) message nor in any [CIM-XML operation](#)  
3041 [request](#) message unless it is a simple operation Request. It shall not be present in any CIM-XML export  
3042 request or response message.

3043 The header identifies the CIM object on which the method is to be invoked using a CIM object path  
 3044 encoded in an [HTTP-safe representation](#). This object shall be a class or instance for an [extrinsic](#) method  
 3045 or a namespace for an [intrinsic](#) method. Firewalls and proxies may use this header to carry out routing  
 3046 and forwarding decisions based on the CIM object that is the target of a method invocation.

3047 `CIMObject = "CIMObject" ":" ObjectPath`

3048

3049 `ObjectPath = CIMObjectPath`

3050 The ObjectPath value is constructed by applying the algorithm defined in 6.3.2 to either of the following  
 3051 child elements within the CIM-XML operation request:

3052 

- The <LOCALNAMESPACEPATH> child element of the <IMETHODCALL> element.

3053 

- The <LOCALCLASSPATH> or <LOCALINSTANCEPATH> child element of the  
 3054 <METHODCALL> element.

3055 If a [WBEM server](#) receives a CIM-XML operation request for which any one of the following statements is  
 3056 true, then it shall fail the request and return a status of "400 Bad Request". Also, it shall include a  
 3057 [CIMError](#) header in the response with a value of `header-mismatch`, subject to the considerations  
 3058 specified in 7.3:

3059 

- The CIMObject header is present, but it has an invalid value.

3060 

- The CIMObject header is not present, but the operation request message is a [Simple Operation](#)  
 3061 [Request](#).

3062 

- The CIMObject header is present, but the operation request message is not a simple operation  
 3063 request.

3064 

- The CIMObject header is present and the operation request message is a simple operation  
 3065 request, but the ObjectPath value does not match the operation request message (where a  
 3066 *match* is defined in 6.3.2).

3067 Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy is not  
 3068 acting on misleading information when it forwards the request based on the content of the CIMObject  
 3069 header. Additional securing of HTTP messages against modification in transit, such as encrypting the  
 3070 payload or appending a digital signature to it, would be required to provide a higher degree of integrity.

### 3071 **6.3.8 CIMExportMethod**

3072 The CIMExportMethod header shall be present in any CIM-XML export request message that contains a  
 3073 simple export request.

3074 This header shall not be present in any CIM-XML export response message nor in any CIM-XML export  
 3075 request message unless it is a simple export request. It shall not be present in any CIM-XML operation  
 3076 request or response message.

3077 The CIMExportMethod header identifies the name of the CIM export method to be invoked, encoded in an  
 3078 HTTP-safe representation. Firewalls and proxies may use this header to carry out routing and forwarding  
 3079 decisions based on the CIM export method to be invoked.

3080 The name of the CIM export method within a simple export request is the value of the NAME attribute of  
 3081 the <EXPMETHODCALL> element.

3082 `CIMExportMethod = "CIMExportMethod" ":" ExportMethodName`

3083

3084 `ExportMethodName = CIMIdentifier`

3085 If a WBEM listener receives a CIM-XML export request for which any one of the following statements is  
3086 true, then it shall fail the request and return a status of "400 Bad Request". Also, it shall include a  
3087 CIMError header in the response with a value of header-mismatch, subject to the considerations specified  
3088 in 7.3:

- 3089 • The CIMExportMethod header is present, but it has an invalid value.
- 3090 • The CIMExportMethod header is not present, but the export request message is a simple export  
3091 request.
- 3092 • The CIMExportMethod header is present, but the export request message is not a simple export  
3093 request.
- 3094 • The CIMExportMethod header is present and the export request message is a simple export  
3095 request, but the CIMIdentifier value (when unencoded) does not match the unique method  
3096 name within the simple export request.

3097 Note that this verification provides a basic level of assurance that any intermediate firewall or proxy is not  
3098 acting on misleading information when it forwards the request based on the content of the  
3099 CIMExportMethod header. Additional securing of HTTP messages against modification in transit, such as  
3100 encrypting the payload or appending a digital signature to it, would be required to provide a higher degree  
3101 of integrity.

### 3102 6.3.9 CIMBatch

3103 The CIMBatch header shall be present in any [CIM-XML operation request](#) message that contains a  
3104 [Multiple Operation Request](#).

3105 This header shall not be present in any [CIM-XML operation response](#) message nor in any [CIM-XML](#)  
3106 [operation request](#) message unless it is a multiple operation request. It shall not be present in any CIM-  
3107 XML export request or response message.

3108 The CIMBatch header identifies the encapsulated operation request message as containing multiple  
3109 method invocations. Firewalls and proxies may use this header to carry out routing and forwarding  
3110 decisions for batched CIM method invocations.

```
3111 CIMBatch = "CIMBatch" ":"
```

3112 If a [WBEM server](#) receives a CIM-XML operation request for which any one of the following statements is  
3113 true, then it must fail the request and return a status of "400 Bad Request". Also it must include a  
3114 [CIMError](#) header in the response with a value of header-mismatch, subject to the considerations  
3115 specified in 7.3:

- 3116 • The CIMBatch header is present, but it has an invalid value.
- 3117 • The CIMBatch header is not present, but the operation request message is a multiple operation  
3118 request.
- 3119 • The CIMBatch header is present, but the operation request message is not a multiple operation  
3120 request.

3121 Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy is not  
3122 acting on misleading information when it forwards the request based on the content of the CIMBatch  
3123 header. Additional securing of HTTP messages against modification in transit, such as encrypting the  
3124 payload or appending a digital signature to it, would be required to provide a higher degree of integrity.

3125 If a WBEM server receives a CIM-XML operation request for which the CIMBatch header is present but  
3126 the server does not support multiple operations, then it shall fail the request and return a status of "501  
3127 Not Implemented". Firewalls or Proxies may also employ this mechanism to compel a [WBEM client](#) to use  
3128 simple operation requests rather than multiple operation requests.

3129 A WBEM client that receives a response of "501 Not Implemented" to a multiple operation request should  
3130 resubmit that request as a series of simple operation requests.

### 3131 6.3.10 CIMExportBatch

3132 The CIMExportBatch header shall be present in any CIM-XML export request message that contains a  
3133 multiple export request.

3134 It shall not be present in any CIM-XML operation request or response message. Also, it shall not be  
3135 present in any CIM-XML export response message nor in any CIM-XML export request message unless it  
3136 is a multiple export request.

3137 The header identifies the encapsulated Export Request Message as containing multiple export method  
3138 invocations. Firewalls and proxies may use this header to carry out routing and forwarding decisions for  
3139 batched CIM Export method invocations.

```
3140     CIMExportBatch = "CIMExportBatch" ":"
```

3141 If a WBEM listener receives a CIM-XML export request for which any one of the following statements is  
3142 true, then it must fail the request and return a status of "400 Bad Request". Also, it must include a  
3143 CIMError header in the response with a value of header-mismatch, subject to the considerations specified  
3144 in [Errors](#):

- 3145 • The CIMExportBatch header is present, but it has an invalid value.
- 3146 • The CIMExportBatch header is not present, but the export request message is a multiple export  
3147 request.
- 3148 • The CIMExportBatch header is present, but the export request message is not a multiple export  
3149 request.

3150 Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy is not  
3151 acting on misleading information when it forwards the request based on the content of the  
3152 CIMExportBatch header. Additional securing of HTTP messages against modification in transit, such as  
3153 encrypting the payload or appending a digital signature to it, would be required to provide a higher degree  
3154 of integrity.

3155 If a WBEM listener receives a CIM-XML export request for which the CIMExportBatch header is present,  
3156 but the WBEM listener does not support multiple exports, then it shall fail the request and return a status  
3157 of "501 Not Implemented". Firewalls or Proxies may also employ this mechanism to compel a WBEM  
3158 client to use simple rather than multiple export requests.

3159 A WBEM client that receives a response of "501 Not Implemented" to a multiple export request should  
3160 resubmit that request as a series of simple export requests.

### 3161 6.3.11 CIMError

3162 The CIMError header may be present in any HTTP response to a CIM-XML message request that is not a  
3163 CIM-XML message response.

3164 It shall not be present in any CIM-XML message response or in any CIM-XML message request.

3165 The CIMError header provides further CIM-specific diagnostic information if the [WBEM server](#) or [WBEM  
3166 listener](#) encounters a fundamental error during processing of the CIM-XML operation request and is  
3167 intended to assist clients to further disambiguate errors with the same HTTP status code:

```
3168     CIMError = "CIMError" ":" cim-error
3169
3170     cim-error = "unsupported-protocol-version" |
```

3171 "multiple-requests-unsupported" |  
3172 "unsupported-cim-version" |  
3173 "unsupported-dtd-version" |  
3174 "request-not-valid" |  
3175 "request-not-well-formed" |  
3176 "request-not-loosely-valid" |  
3177 "header-mismatch" |  
3178 "unsupported-operation"

### 3179 6.3.12 CIMRoleAuthenticate

3180 A WBEM server may return a CIMRoleAuthenticate header as part of the 401 Unauthorized response  
3181 along with the WWW-Authenticate header. The CIMRoleAuthenticate header must meet the challenge of  
3182 indicating the WBEM server policy on role credentials.

3183 challenge = "credentialrequired" | "credentialoptional" | "credentialnotrequired"

- 3184 • A challenge of `credentialrequired` indicates that the WBEM server requires that a WBEM  
3185 client must present a credential if it seeks to assume a role.
- 3186 • A challenge of `credentialoptional` indicates that the credential is optional. If a credential is  
3187 not sent, the WBEM server allows the role assumption if it is permitted for the given user.  
3188 However, certain operations that require the role credential may not succeed.
- 3189 • A challenge of `credentialnotrequired` indicates that no credential is required to assume  
3190 the role.

3191 Absence of the CIMRoleAuthenticate header indicates that the WBEM server does not support role  
3192 assumption. A WBEM client should handle each of these cases appropriately.

3193 The challenge does not contain any authorization scheme, realm, or other information. A WBEM client  
3194 should extract this information from the WWW-Authenticate header. This implies that for any given  
3195 request, the role credentials should use the same scheme as those required for the user credentials.

3196 A WBEM server allows role assumption to succeed only if the user is allowed to assume the role.  
3197 Therefore, even if appropriate credentials are presented, role assumption can fail. If either the user  
3198 authentication or role assumption fails, the entire authentication operation fails.

3199 To maintain backward compatibility, a WBEM server that supports role assumption must allow user  
3200 authentication even if no role is specified.

### 3201 6.3.13 CIMRoleAuthorization

3202 The CIMRoleAuthorization header is supplied along with the normal authorization header that the WBEM  
3203 client populates to perform user authentication. If the WBEM client needs to perform role assumption and  
3204 the WBEM server challenge is `credentialrequired`, the CIMRoleAuthorization header must be supplied  
3205 with the appropriate credentials. The credentials supplied as part of the CIMRoleAuthorization header  
3206 must use the same scheme as those specified for the authorization header, as specified in [RFC2617](#).  
3207 Therefore, both Basic and Digest authentication are possible for the role credential.

3208 If the WBEM client wishes to assume a role but does not wish to supply role credentials for server  
3209 challenge `credentialoptional` or `credentialnotrequired`, the CIMRoleAuthorization header must set the  
3210 `auth-scheme` field as specified in [RFC2617](#) to be "role". The `auth-param` must contain the role name.

3211 A WBEM server that supports roles must be capable of handling the presence of credentials in the  
3212 CIMRoleAuthorization header (that is `auth-scheme` not set to "role") regardless of whether it is expecting  
3213 credentials or not. It may choose to ignore these credentials.

### 3214 6.3.14 CIMStatusCodeDescription

3215 If a CIM product includes the CIMStatusCode trailer, it may also include the CIMStatusCodeDescription  
3216 trailer. The value of this trailer is a string describing the nature of the error. A CIM product shall not  
3217 include this trailer if the CIMStatusCode trailer is not present.

### 3218 6.3.15 WBEMServerResponseTime

3219 The WBEMServerResponseTime header may be present in any CIM response message. If it is present,  
3220 the header shall contain a measure, specified in microseconds, of the elapsed time required by the  
3221 WBEM server to process the request and create a response. Specifically, WBEMServerResponseTime  
3222 describes the time elapsed since the WBEM server received the CIM request message and the  
3223 associated CIM response message was ready to send to the WBEM client.

3224 WBEMServerResponseTime = "WBEMServerResponseTime" ":", where the response time must be  
3225 representable as a 64-bit unsigned integer value. If the actual elapsed time exceeds the maximum  
3226 representable value, then the maximum value shall be returned. If the actual elapsed time is less than 1  
3227 microsecond, then a 0 shall be returned.

3228 Although a WBEM client may ignore the WBEMServerResponseTime header, it shall allow this header to  
3229 be included in a response.

## 3230 7 HTTP Requirements and Usage

3231 This clause describes HTTP support and the use of standard headers.

### 3232 7.1 HTTP and HTTPS Support

3233 CIM products shall support CIM-XML messages in HTTP. The following applies to this case:

- 3234 • CIM products should support HTTP/1.1 as defined in [RFC2616](#).

---

### 3235 DEPRECATED

3236 CIM products may support HTTP/1.0 as defined in [RFC1945](#).

- 3237 • Support for HTTP/1.0 is deprecated since version 1.4 of this document; HTTP/1.1 should be  
3238 supported instead.

---

### 3239 DEPRECATED

3240 CIM products should support CIM-XML messages in HTTPS. If they do, the following applies to this case:

- 3241 • CIM products shall support HTTPS as defined in [RFC2818](#). This includes the use of HTTP  
3242 within HTTPS, as defined in [RFC2818](#).

3243 NOTE [RFC2818](#) describes the use of TLS 1.0 and higher but not the use of SSL 2.0 or 3.0.

- 3244 • Within their support of HTTPS, CIM products:

- 3245 – shall support TLS 1.0 (also known as SSL 3.1) as defined in [RFC2246](#). Note that TLS 1.0  
3246 implementations may be vulnerable when using CBC cipher suites

- 3247 – should support TLS 1.1 as defined in [RFC4346](#)

- 3248 – should support TLS 1.2 as defined in [RFC5246](#)

- 3249 – should not support [SSL 2.0](#) or [SSL 3.0](#) because of known security issues in these versions

3250 NOTE [RFC5246](#) describes in Appendix E "Backward Compatibility" how the secure sockets layer can  
3251 be negotiated.

3252 Requirements and considerations for authentication and encryption between CIM products are described  
3253 in 7.4.

3254 CIM products that use extension headers as defined in this document shall conform to the requirements  
3255 defined in [RFC2774](#) for their use.

## 3256 7.2 Use of Standard HTTP Headers

3257 Unless otherwise stated in this document, CIM products shall comply with the requirements on the use of  
3258 standard HTTP headers described in [RFC1945](#) and [RFC2616](#). This clause defines only *additional*  
3259 requirements on CIM products with respect to the use of these standard HTTP headers in a CIM-XML  
3260 message.

3261 Note that CIM products should not use HTTP headers defined in [RFC2068](#) but deprecated in [RFC2616](#)  
3262 (for example, Public, Content-Base).

### 3263 7.2.1 Accept

3264 If a [WBEM client](#) includes an Accept header in a request, it shall specify a value that allows the WBEM  
3265 server to return an entity body of "text/xml" or "application/xml" in the response.

3266 A [WBEM server](#) or [WBEM listener](#) shall accept any value for this header stating that "text/xml" or  
3267 "application/xml" is an acceptable type for a response entity. A WBEM server or WBEM listener should  
3268 return "406 Not Acceptable" if the Accept header indicates that neither of these content types is  
3269 acceptable.

3270 If a WBEM server or WBEM listener accepts a request to return an entity of a type other than "text/xml" or  
3271 "application/xml", the nature of the response is outside the scope of this document.

### 3272 7.2.2 Accept-Charset

3273 If a [WBEM client](#) includes an Accept-Charset header in a request, it shall specify a value that allows the  
3274 WBEM server or WBEM listener to return an entity body using the character set "UTF-8".

3275 A [WBEM server](#) or [WBEM listener](#) shall accept any value for this header asserting that "UTF-8" is an  
3276 acceptable character set for a response entity. If the client does not provide an Accept-Charset, then  
3277 "UTF-8" should be assumed by the [WBEM server](#) or [WBEM listener](#).

3278 `Accept-Charset: UTF-8`

3279 A WBEM server or WBEM listener shall return "406 Not Acceptable" if the character set requested in the  
3280 Accept-Charset header is not supported.

3281 If a WBEM server or WBEM listener accepts a request to return an entity using a character set other than  
3282 "UTF-8", the behavior of the subsequent WBEM client and WBEM server interaction is outside the scope  
3283 of this document. See 7.8 for details.

### 3284 7.2.3 Accept-Encoding

3285 If a [WBEM client](#) includes an Accept-Encoding header in a request, it shall specify a q value that allows  
3286 the WBEM server or WBEM listener to use the "Identity" encoding. The value shall be greater than 0 or  
3287 not specified.

3288 `Accept-Encoding: Identity`

3289 `Accept-Encoding: Identity; q=1.0`

3290 A [WBEM server](#) or [WBEM listener](#) shall accept any value for this header asserting that "Identity" is an  
3291 acceptable encoding for the response entity.

3292 A WBEM server or WBEM listener shall return "406 Not Acceptable" if the Accept-Encoding header  
3293 indicates that the requested encoding is not acceptable.

#### 3294 **7.2.4 Accept-Language**

3295 If a WBEM client includes an Accept-Language header in a request, it shall request a language-range,  
3296 special-range, or both. The WBEM client shall also allow any language to be returned if the requested  
3297 languages cannot be supported. This is accomplished by including the special-range, "\*". The WBEM  
3298 client may request multiple languages. Each language has equal priority, unless a q value is provided.

3299 `Accept-Language: zh, *`

3300 `Accept-Language: zh;q=1.0, en;q=.7, *`

3301 Each CIM element in the response should be localized in only one language. A CIM element shall not be  
3302 duplicated in the response because it is localized in more than one language.

3303 WBEM servers may support multiple languages. A CIM product shall interpret the use of the special-  
3304 range value, "\*", as a request to return the response content using the default language defined for the  
3305 target processing the request. Multiple targets, with different default language settings, may participate in  
3306 the construction of a response. (See [RFC2616](#) section 3.10 and [ISO 639-1](#).)

3307 See 7.8 for more information.

#### 3308 **7.2.5 Accept-Ranges**

3309 [WBEM clients](#) shall not include the Accept-Ranges header in a request. A [WBEM server](#) or [WBEM](#)  
3310 [listener](#) shall reject a request that includes an Accept-Range header with a status of "406 Not  
3311 Acceptable".

#### 3312 **7.2.6 Allow**

3313 If a [WBEM server](#) or [WBEM listener](#) is returning a "405 Method Not Allowed" response to a CIM-XML  
3314 message request, then the Allow header shall include either M-POST or POST. Whether it includes any  
3315 other HTTP methods is outside the scope of this document.

#### 3316 **7.2.7 Authorization**

3317 See 7.4 for details.

#### 3318 **7.2.8 Cache-Control**

3319 Generally, a CIM-XML message request may consist of a mixture of CIM method invocations, some of  
3320 which may be eminently able to cache (for example, the manufacturer label on a disk drive) and some of  
3321 which may be decidedly impossible to cache (for example, format a disk drive).

3322 Furthermore, the encapsulation of such multiple method invocations in an HTTP POST or M-POST  
3323 means that if a CIM-XML message request has any effect on an HTTP cache it is likely to be one of  
3324 invalidating cached responses for the target WBEM server or WBEM listener. Indeed, [HTTP/1.1](#) stipulates  
3325 that by default POST responses cannot be cached unless the WBEM server indicates otherwise using an  
3326 appropriate Cache-Control or Expires header.

3327 For these reasons, CIM-XML message responses should not be considered as able to be cached. A  
3328 [WBEM server](#) or [WBEM listener](#) should not include a Cache-Control header in a CIM-XML message  
3329 response that might indicate to a cache that the response can be cached.

3330 If the WBEM server or WBEM listener is responding to a CIM-XML message request conveyed in an M-  
 3331 POST request, then in accordance with [RFC2774](#) the WBEM server or WBEM listener shall include a no-  
 3332 cache control directive to prevent inadvertent caching of the "Ext" header, as in the following example:

3333 EXAMPLE

```
3334     HTTP/1.1 200 OK
3335     Ext:
3336     Cache-Control: no-cache
3337     ...
```

### 3338 7.2.9 Connection

3339 The following courses of action are recommended for connections:

- 3340 • [WBEM clients](#) should avoid the use of the "Connection: close" header unless it is known in  
 3341 advance that this is the only request likely to be sent out on that connection.
- 3342 • [WBEM servers](#) and [WBEM listener](#) support persistent connections wherever possible.

3343 Timeout mechanisms should be employed to remove idle connections on the WBEM client, WBEM  
 3344 server, and WBEM listener. The details of timeout mechanisms are outside the scope of this document.  
 3345 Clients should be cautious in retrying requests, especially if they are not idempotent (for example, method  
 3346 invocation).

3347 WBEM clients, WBEM servers, and WBEM listeners should support pipelining (HTTP/1.1 only, see  
 3348 [RFC2616](#)) if possible, but be aware of the requirements defined in [RFC2616](#). In particular, attention is  
 3349 drawn to the requirement from [RFC2616](#) that clients not pipeline requests using non-idempotent methods  
 3350 or non-idempotent sequences of methods. A client that needs to send a non-idempotent request should  
 3351 wait to send that request until it receives the response status for the previous request.

### 3352 7.2.10 Content-Encoding

3353 If a [WBEM client](#) includes a Content-Encoding header in a request, it should specify a value of "identity",  
 3354 unless there is good reason to believe that the WBEM server or WBEM listener can accept another  
 3355 encoding.

### 3356 7.2.11 Content-Language

3357 The Content-Language entity-header field of a CIM-XML message describes the natural language(s) of  
 3358 the intended audience of the content.

3359 A CIM-XML message may contain a Content-Language header. The value of the Content-Language  
 3360 header in a CIM response message shall be consistent with the Accept-Language values specified in the  
 3361 corresponding CIM request message. If the WBEM server cannot determine one or more of the content  
 3362 languages used to construct the response, then the Content-Language entity shall not be returned.

3363 Multiple targets using different Content-Language values may participate in constructing a response. The  
 3364 Content-Language field shall reflect all Content-Language values used to construct the response. The  
 3365 content of a CIM-XML message may contain elements in languages not listed in the Content-Language  
 3366 field.

```
3367     Content-Language: en
```

3368 See 7.8 for details.

### 3369 7.2.12 Content-Range

3370 [WBEM clients](#), [WBEM servers](#), and [WBEM listeners](#) shall not use this header.

### 3371 7.2.13 Content-Type

3372 [WBEM clients](#), [WBEM servers](#), and [WBEM listeners](#) shall specify (and accept) a media type for the  
3373 Content-Type header of either "text/xml" or "application/xml" as defined in [RFC2376](#). In addition, they  
3374 may specify and shall accept a "charset" parameter as defined in [RFC2616](#). If a "charset" parameter is  
3375 specified, it shall have the value "utf-8" either with or without surrounding double quotes. The sending  
3376 side should use the form without double quotes. The receiving side shall support both forms. If a "charset"  
3377 parameter is not specified, the receiving side shall assume "utf-8" as a default.

3378 Examples of valid Content-Type headers are:

```
3379 Content-type: text/xml
3380 Content-type: text/xml; charset=utf-8
3381 Content-type: text/xml; charset="utf-8"
3382 Content-type: application/xml
3383 Content-type: application/xml; charset=utf-8
3384 Content-type: application/xml; charset="utf-8"
```

### 3385 7.2.14 Expires

3386 For the reasons described in 7.2.8, a [WBEM server](#) or [WBEM listener](#) shall not include an Expires header  
3387 in a CIM-XML message response that might indicate to a cache that the response can be cached.

### 3388 7.2.15 If-Range

3389 [WBEM clients](#), [WBEM servers](#), and [WBEM listeners](#) shall not use this header.

### 3390 7.2.16 Proxy-Authenticate

3391 See 7.4 for details.

### 3392 7.2.17 Range

3393 [WBEM clients](#), [WBEM servers](#), and [WBEM listeners](#) shall not use this header.

### 3394 7.2.18 WWW-Authenticate

3395 See 7.4 for details.

## 3396 7.3 Errors and Status Codes

3397 This clause defines how [WBEM servers](#) and [WBEM listeners](#) shall handle errors that occur in processing  
3398 a CIM-XML message request. This document does not introduce any new HTTP response status codes.

3399 If there is an error in processing the HTTP Request-Line or standard HTTP headers, the WBEM server or  
3400 WBEM listener shall take appropriate action as dictated by its conformance to the relevant version of  
3401 HTTP (see 7.1).

3402 Otherwise, if there are any mandatory extension declarations that the WBEM server does not support it  
3403 shall respond with a "510 Not Extended" status according to [RFC2774](#).

3404 Otherwise, the request shall be processed in accordance with the relevant version of HTTP (see 7.1) and  
3405 the additional rules defined in this document.

3406 Assuming that the HTTP request is otherwise correct, the WBEM server or WBEM listener shall use the  
3407 following status codes when processing the CIM extension headers:

- 3408 • 501 Not Implemented

- 3409 This status code indicates that one of the following situations occurred:
- 3410 – The [CIMProtocolVersion](#) extension header in the request specifies a version of the CIM  
3411 mapping onto HTTP that is not supported by this WBEM server or WBEM listener. The  
3412 WBEM server or WBEM listener shall include a [CIMError](#) header in the response with a  
3413 value of `unsupported-protocol-version`.
  - 3414 – The client specified a [Multiple Operation Request](#) (or multiple Export Request), and the  
3415 WBEM server (or WBEM listener) does not support such requests. The WBEM server or  
3416 WBEM listener shall include a [CIMError](#) header in the response with a value of  
3417 `multiple-requests-unsupported`.
  - 3418 – The CIMVERSION attribute in the message request is not set to a proper value. The  
3419 CIMVERSION attribute shall be in the form of "M.N", where M is the major revision of the  
3420 specification in numeric form and N is the minor revision in numeric form. The version shall  
3421 be at "2.0" or greater (for example, "2.0" or "2.3"). The WBEM server or WBEM listener  
3422 shall include a CIMError header in the response with a value of `unsupported-cim-`  
3423 `version`.
  - 3424 – The DTDVERSION attribute in the message request is not set to a proper value. The  
3425 DTDVERSION attribute shall be in the form of "M.N", where M is the major revision of the  
3426 specification in numeric form and N is the minor revision in numeric form. The version shall  
3427 be at "2.0" or greater (for example, "2.0" or "2.1"). The WBEM server or WBEM listener  
3428 shall include a CIMError header in the response with a value of `unsupported-dtd-`  
3429 `version`.
- 3430 • 401 Unauthorized
- 3431 The WBEM server or WBEM listener is configured to require that a client authenticate itself  
3432 before it can issue CIM-XML message requests to the WBEM server or WBEM listener.
- 3433 • 403 Forbidden
- 3434 The WBEM server or WBEM listener does not allow the client to issue CIM-XML message  
3435 requests. The WBEM server or WBEM listener may alternatively respond with a "404 Not  
3436 Found" if it does not wish to reveal this information to the client.
- 3437 • 407 Proxy Authentication Required
- 3438 The WBEM server or WBEM listener is configured to require that the proxy authenticate itself  
3439 before it can issue CIM-XML message requests on behalf of a WBEM client to the WBEM  
3440 server or WBEM listener.
- 3441 Assuming that the CIM extension headers are correct, a validating WBEM server or WBEM listener (one  
3442 that enforces the validity of the CIM-XML message request with respect to the CIM XML DTD) shall use  
3443 the following status code when processing the entity body containing the CIM-XML message request:
- 3444 • 400 Bad Request
- 3445 The entity body defining the CIM-XML message request is not well-formed or not valid with  
3446 respect to the CIM XML DTD. The WBEM server or WBEM listener shall include a CIMError  
3447 header in the response with a value of `request-not-well-formed` or `request-not-`  
3448 `valid` (as appropriate).
- 3449 A loosely-validating WBEM server or WBEM listener only enforces the CIM-XML message request to be  
3450 [loosely valid](#). Therefore, it may reject a CIM-XML message request that is not loosely valid with an HTTP  
3451 status code of 400 (Bad Request) before further processing. In this case, the WBEM server or WBEM  
3452 listener shall include a [CIMError](#) header in the response with a value of `request-not-loosely-`  
3453 `valid`.

- 3454 A loosely-validating WBEM server or WBEM listener shall reject a CIM-XML message request that is not  
3455 well-formed with an HTTP status code of 400 (Bad Request). In this case, the WBEM server or WBEM  
3456 listener shall include a [CIMError](#) header in the response with a value of `request-not-well-formed`.
- 3457 A loosely-validating WBEM server or WBEM listener shall not reject an invalid CIM-XML message request  
3458 that is loosely valid in the XML sense.
- 3459 A loosely-validating WBEM server or WBEM listener shall ultimately signal an error to the WBEM client if  
3460 the CIM-XML message request is not loosely valid. That is, the request is missing required content or the  
3461 required content is incorrect, such as an attribute with an invalid value according to the CIM XML DTD. It  
3462 is not mandated to reject a CIM-XML message request before processing, for to do otherwise would  
3463 compel the WBEM server or WBEM listener to check the complete request before processing can begin  
3464 and this would be as expensive as requiring the WBEM server or WBEM listener to fully validate the  
3465 request. Therefore, a loosely-validating server or listener may elect to begin processing the request and  
3466 issuing a response (with an HTTP success status code) before verifying that the entire request is loosely  
3467 valid.
- 3468 A WBEM client may use the [CIMValidation](#) header mechanism to determine whether a WBEM server or  
3469 WBEM listener is validating or loosely-validating.
- 3470 Assuming that the CIM-XML message request is correctly formed as previously described, the WBEM  
3471 server or WBEM listener shall process the request accordingly and return a CIM-XML message response.
- 3472 The entity body shall be a correct CIM-XML message response for that request.
- 3473 If the CIM-XML message response contains an entity that is a simple message response, then the  
3474 response status shall be "200 OK". Otherwise, the response status shall be "207 Multistatus".

## 3475 7.4 Security Considerations

- 3476 This subclause describes requirements and considerations for authentication and message encryption  
3477 between CIM products.

### 3478 7.4.1 Authentication

- 3479 This subclause describes requirements and considerations for authentication between CIM products.  
3480 Specifically, authentication happens from WBEM clients to WBEM servers for CIM-XML operation  
3481 messages, and from WBEM servers to WBEM listeners for CIM-XML export messages. The  
3482 authentication mechanisms defined in this subclause apply to both HTTP and HTTPS.
- 3483 CIM products may support operating without the use of authentication. This practice is not recommended  
3484 and should only be done in environments where lack of network privacy is not an issue (for example, in a  
3485 physically secure private network or on the same operating system).
- 3486 Basic authentication is described in [RFC1945](#) and [RFC2068](#). Digest authentication is defined in  
3487 [RFC2069](#). Both authentication schemes are covered in a consolidated document ([RFC2617](#)), which also  
3488 makes a number of improvements to the original specification of digest authentication. This document  
3489 requires conformance to [RFC2617](#) but not to the earlier documents.
- 3490 Basic authentication provides a very rudimentary level of authentication, with the major weakness that the  
3491 client password is sent over the wire in unencrypted form (unless HTTPS is used)..
- 3492 CIM products may support basic authentication as defined in [RFC2617](#). Basic authentication without  
3493 HTTPS should only be used in environments where lack of network privacy is not an issue.
- 3494 Digest authentication verifies that both parties share a common secret without having to send that secret.
- 3495 CIM products should support digest authentication as defined in [RFC2617](#).

3496 CIM products may support authentication mechanisms not covered by [RFC2617](#). One example are public  
3497 key certificates as defined in [X.509](#).

3498 WBEM servers and WBEM listeners should require that WBEM clients and WBEM servers, respectively,  
3499 authenticate themselves. This document does not mandate this because it is recognized that in some  
3500 circumstances the WBEM server or WBEM listener may not require or wish the overhead of employing  
3501 authentication. WBEM servers and WBEM listeners should carefully consider the performance/security  
3502 tradeoffs in determining how often to issue challenges to WBEM clients and WBEM servers, respectively.

3503 A WBEM server or WBEM listener that returns a "401 Unauthorized" response to a CIM message request  
3504 shall include one WWW-Authenticate response-header indicating one supported authentication  
3505 mechanism. This document does not mandate use of basic or digest authentication because it is  
3506 recognized that in some circumstances the WBEM server or WBEM listener may use bespoke  
3507 authentication mechanisms not covered by [RFC2617](#). Similar considerations apply to the use of the  
3508 Proxy-Authenticate response-header in "407 Proxy Authentication Required".

## 3509 7.4.2 Message Encryption

3510 Encryption of messages between CIM products is supported by the use of HTTPS in the communication  
3511 between CIM products. Requirements for the use of HTTPS and its underlying secure sockets are  
3512 defined in 7.1.

3513 The following requirements on cipher suites apply to CIM products that support HTTPS:

- 3514 • The TLS\_DHE\_DSS\_WITH\_3DES\_EDE\_CBC\_SHA cipher suite (hexadecimal value 0x0013)  
3515 shall be supported when using TLS 1.0. Note that [RFC2246](#) defines this cipher suite to be  
3516 mandatory for TLS 1.0
- 3517 • The TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA cipher suite (hexadecimal value 0x000A) shall  
3518 be supported when using TLS 1.1. Note that [RFC4346](#) defines this cipher suite to be mandatory  
3519 for TLS 1.1
- 3520 • The TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA cipher suite (hexadecimal value 0x002F) shall be  
3521 supported when using TLS 1.2. Note that [RFC5246](#) defines this cipher suite to be mandatory for  
3522 TLS 1.2
- 3523 • The TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 cipher suite (hexadecimal value 0x003C)  
3524 should be supported when using TLS 1.2, in order to meet the transition to a security strength of  
3525 112 bits (guidance is provided in [NIST 800-57](#) and [NIST 800-131A](#))
- 3526 • Any additional cipher suites may be supported

## 3527 7.5 Determining WBEM server Capabilities

3528 If a WBEM server can return capabilities information, there are two techniques for returning this  
3529 information as defined in this document:

- 3530 • The preferred technique is through the use of the classes defined in 7.5.1.
- 3531 • Alternatively, use of the HTTP OPTIONS method as defined in 7.5.2 is allowed because  
3532 historically it is the original technique defined for requesting capabilities information.

3533 Use of the CIM classes defined in 7.5.1 is strongly encouraged and it is expected that this method will be  
3534 enhanced and extended in the future to provide more capabilities information. The future use of the HTTP  
3535 OPTIONS method to determine capabilities of WBEM servers is discouraged. It will probably not be  
3536 expanded significantly and may be reviewed for possible deprecation in the next major revision of this  
3537 document.

### 3538 7.5.1 Determining WBEM server Capabilities through CIM Classes

3539 A set of CIM classes is defined specifically to return WBEM server capabilities information as follows:

- 3540 • CIM\_ObjectManager

3541 This class is a type of CIM\_Service that defines the capabilities of the target WBEM server.

- 3542 • CIM\_ObjectManagerCommunicationMechanism

3543 This class describes access to the target WBEM server. It defines the capabilities of the WBEM  
3544 server that are available through the target Object Manager Communication mechanism. A  
3545 WBEM server is allowed to support different capabilities through different communication  
3546 mechanisms.

- 3547 • CIM\_CIMXMLCommunicationMechanism

3548 This class specializes on ObjectManagerCommunicationMechanism, adding properties specific  
3549 to the CIM-XML encoding and protocol.

- 3550 • CIM\_CommMechanismForManager

3551 This association between CIM\_ObjectManager and  
3552 CIM\_ObjectManagerCommunicationMechanism defines the communications protocols (and  
3553 corresponding capabilities) available on the target WBEM server through the  
3554 ObjectManagerCommunicationMechanism instances.

3555 A WBEM client may use instances of these CIM classes to determine the CIM capabilities (if any) of the  
3556 target WBEM server. A WBEM server that supports capabilities determination through these classes shall  
3557 support at least the Enumerate Instance and Get Instance operations for the classes. The use of other  
3558 methods of the basic read profile is optional. A WBEM server that does not support the determination of  
3559 CIM capabilities through these classes shall return [CIM\\_ERR\\_NOT\\_FOUND](#) to any instance or class  
3560 request on these classes. These classes shall not be used for reporting any other information than  
3561 capabilities of the target WBEM server.

3562 To provide interoperability, the CIM object manager classes shall exist in a well-known namespace.  
3563 Because there is no discovery mechanism that can define this well-known namespace to a WBEM client,  
3564 it shall be one or more predefined namespaces. Therefore, to ensure interoperability, we recommend that  
3565 pending future extensions of the WBEM specifications include discovery tools that define a namespace  
3566 for these classes in a WBEM server; these predefined namespaces should exist in either the root  
3567 namespace or in the /root/CIMV2 namespace.

3568 A WBEM server that supports capabilities reporting through these classes shall correctly report the  
3569 current actual capabilities of the target WBEM server and shall report on all capabilities defined. A WBEM  
3570 server is allowed to report "none" if the capability does not exist or "unknown" if the status of the capability  
3571 is unknown at the time of the request for those properties where these choices exist in the properties  
3572 definition. Because the CIM\_ObjectManager object provides information on the target WBEM server, only  
3573 a single instance of this class may exist in a WBEM server.

3574 The capabilities to be reported through the CIM\_ObjectManagerCommunicationMechanism are as  
3575 follows:

- 3576 • CommunicationMechanism property, which defines the communication protocol for the  
3577 CommunicationMechanism object. A compliant WBEM server shall include the CIM-XML  
3578 protocol for at least one ObjectManagerCommunicationMechanism instance.

- 3579 • ProfilesSupported property, which defines the functional profiles supported as defined in clause  
3580 5.4.4. All WBEM servers shall support the basic-read functional group. All WBEM clients may  
3581 assume that any WBEM server supports the basic-read functional group. The list of functional  
3582 groups returned by a WBEM server shall contain the basic-read group and shall not contain

3583 duplicates. WBEM clients shall ignore duplicate entries in the functional-group list. If a functional  
 3584 group is included in the list, the WBEM client shall assume that all other groups on which it  
 3585 depends (according to the rules defined in 5.4.4) are also supported. A WBEM server should  
 3586 not explicitly include a functional group in the list whose presence may be inferred implicitly by a  
 3587 dependency. Support for a functional group does not imply that any method from that group will  
 3588 always succeed. Rather, the absence of the functional group from this list (whether explicit or  
 3589 implied) indicates to the WBEM client that methods in that group will never succeed.

3590 • `MultipleOperationsSupported` property, which defines whether the target WBEM server supports  
 3591 multiple operation requests as defined in 5.4.2. `True` in this property indicates that the WBEM  
 3592 server can accept and process multiple operation requests. `False` indicates that the WBEM  
 3593 server can accept only single operation requests.

3594 • `AuthenticationMechanismsSupported` property, which defines the authentication mechanisms  
 3595 supported by the target WBEM server as defined in 7.4.

3596 • `PulledEnumerationClosureOnExceedingServerLimits` property, which indicates whether the  
 3597 WBEM server supports closure of Pulled Enumeration sessions based upon exceeding server  
 3598 limits.

3599 • `PulledEnumerationContinuationOnErrorSupported` property, which indicates whether the WBEM  
 3600 server supports continuation on error for Pulled enumerations.

3601 • `PulledEnumerationMinimumOperationTimeout` (`PulledEnumerationMaximumOperationTimeout`)  
 3602 property, which indicates the minimum (maximum) operation timeout allowed by the WBEM  
 3603 server for Pulled enumerations.

3604 Compliant WBEM servers may report additional capabilities for the `CommunicationMechanism Functional`  
 3605 `Profiles`, `QueryLanguageSupported`, and `AuthenticationMechanismSupported` by defining the "other"  
 3606 enumeration in the property and returning additional information in the associated "additional capabilities"  
 3607 property.

## 3608 7.5.2 Determining WBEM server Capabilities through the HTTP Options

3609 A WBEM client may use the `OPTIONS` method to determine the CIM capabilities (if any) of the target  
 3610 server. A [WBEM server](#) may support the `OPTIONS` method (for example, WBEM servers supporting only  
 3611 HTTP/1.0 would not support `OPTIONS`).

3612 To support the ability for a WBEM server to declare its CIM capabilities independently of HTTP, the DMTF  
 3613 intends to publish a CIM schema (in a separate document) describing such capabilities. In particular, this  
 3614 mechanism would allow servers that do not support the `OPTIONS` method to declare their capabilities to  
 3615 a client.

3616 If a WBEM server supports the `OPTIONS` method, it should return the following headers in the response:

3617 • CIM Extension Header [CIMProtocolVersion](#), which provides a way for a client to discover the  
 3618 version of the CIM HTTP mapping supported by the WBEM server.

3619 • CIM Extension Header `CIMSupportedFunctionalGroups`, which provides a way for a client to  
 3620 discover the CIM operations supported by the WBEM server.

3621 • CIM Extension Header [CIMSupportsMultipleOperations](#), which provides a way for the client to  
 3622 discover whether the WBEM server can support [Multiple Operation Requests](#).

3623 In addition, if the WBEM server supports one or more query languages for the `ExecQuery` operation (see  
 3624 5.4.2.13), it should return the following header in the response:

3625 • CIM Extension Header [CIMSupportedQueryLanguages](#), which allows the client to discover the  
 3626 query languages supported by the WBEM server for the `ExecQuery` operation.

3627 In addition, if the WBEM server runs in a fixed validation mode, it should return the following header in the  
3628 response:

- 3629 • CIM Extension Header [CIMValidation](#), which allows the client to determine whether the WBEM  
3630 server is strictly validating or loosely validating.

3631 If the [CIMProtocolVersion](#), [CIMSupportedFunctionalGroups](#), [CIMSupportsMultipleOperations](#),  
3632 [CIMValidation](#), or [CIMSupportedQueryLanguages](#) extension headers are included in the response, the  
3633 WBEM server shall declare them as optional extension headers using the "Opt" header defined in  
3634 [RFC2774](#).

3635 The full format of the "Opt" header declaration for this document is:

```
3636 Opt          = "Opt" ":" "http://www.dmtf.org/cim/mapping/http/v1.0"
3637              ";" "ns" "=" header-prefix
3638
3639 header-prefix = 2*DIGIT
```

3640 This header-prefix should be generated at random on a per-HTTP message basis and should not  
3641 necessarily be a specific number.

3642 EXAMPLE: The following is a fragment of a legitimate OPTIONS response from a WBEM server:

```
3643 HTTP/1.1 200 OK
3644 Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=77
3645 77-CIMProtocolVersion: 1.0
3646 77-CIMSupportedFunctionalGroups: basic-read
3647 77-CIMBatch
3648 77-CIMSupportedQueryLanguages: wql
3649 ...
```

### 3650 7.5.2.1 CIMSupportedFunctionalGroups

3651 The CIMSupportedFunctionalGroups extension header should be returned by a [WBEM server](#) in any  
3652 OPTIONS response. It shall not be returned in any other scenario.

3653 This header is defined as follows:

```
3654 CIMSupportedFunctionalGroups = "CIMSupportedFunctionalGroups" ":"
3655                               1#functional-group
3656
3657 functional-group = "basic-read" |
3658                  "basic-write" |
3659                  "schema-manipulation" |
3660                  "instance-manipulation" |
3661                  "qualifier-declaration" |
3662                  "association-traversal" |
3663                  "query-execution"
```

3664 The functional group definitions correspond directly to those listed in 5.5.3. All WBEM servers shall  
3665 support the basic-read functional group. All [WBEM clients](#) may assume that any WBEM server supports  
3666 the basic-read functional group.

3667 The list of functional groups returned by a WBEM server shall contain the basic-read group and shall not  
3668 contain any duplicates. WBEM clients shall ignore any duplicate entries in the functional-group list.

3669 If a functional group is included in the list, the WBEM client shall assume that all other groups on which it  
 3670 depends (according to the rules defined in 5.5.3) are also supported. A WBEM server should not explicitly  
 3671 include a functional group in the list if the presence of the group may be implied by a dependency.

3672 EXAMPLE: The following HTTP response message indicates that the WBEM server supports instance-  
 3673 manipulation, association-traversal, basic-write, and basic-read.

```
3674 HTTP/1.1 200 OK
3675 Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=77
3676 77-CIMProtocolVersion: 1.0
3677 77-CIMSupportedFunctionalGroups: association-traversal, instance-manipulation
3678 ...
```

3679 Support for a functional group does *not* imply that any method from that group will always succeed.  
 3680 Rather, the absence (whether explicit or implied) of the functional group from this header is an indication  
 3681 to the WBEM client that methods in that group will *never* succeed.

### 3682 7.5.2.2 CIMSupportsMultipleOperations

3683 The CIMSupportsMultipleOperations extension header shall be returned in an OPTIONS response by any  
 3684 [WBEM server](#) that supports [Multiple Operation Requests](#). It shall not be returned in any other  
 3685 circumstances.

3686 This header is defined as follows:

```
3687 CIMSupportsMultipleOperations = "CIMSupportsMultipleOperations"
```

3688 The presence of this header indicates that the WBEM server can accept and process multiple operation  
 3689 requests. The absence of this header indicates that the WBEM server can only accept and process  
 3690 [Simple Operation Requests](#).

### 3691 7.5.2.3 CIMSupportedQueryLanguages (DEPRECATED)

3692 The CIMSupportedQueryLanguages extension header identifies the query languages supported by the  
 3693 WBEM server for the ExecQuery operation (see 5.4.2.13).

3694 **DEPRECATION NOTE:** The CIMSupportedQueryLanguages extension header has been deprecated in  
 3695 version 1.4 of this document, because it was used only for the ExecQuery operation.

3696 The CIMSupportedQueryLanguages extension header should be returned in any OPTIONS response by  
 3697 a [WBEM server](#) that supports at least one such query language. It shall not be returned in any other  
 3698 scenario.

3699 This header is defined as follows (token has the meaning conferred by [RFC1945](#) and [RFC2616](#)):

```
3700 CIMSupportedQueryLanguages = "CIMSupportedQueryLanguages" ":" 1#query-language
3701
3702 query-language = token
```

3703 The `query-language` value shall be treated as case-insensitive. It is anticipated that query languages  
 3704 will be submitted for approval to the DMTF, and each submission will define a value for this token to  
 3705 enable it to be specified in this header.

### 3706 7.5.2.4 CIMValidation

3707 The CIMValidation extension header may be returned by a [WBEM server](#) to provide information about the  
 3708 level of validation of [CIM-XML operation request](#) messages.

3709 This header is defined as follows:

```

3710     CIMValidation = "CIMValidation" ":" validation-level
3711
3712     validation-level = "validating" | "loosely-validating"

```

3713 A validation-level of `validating` indicates that the WBEM server always applies strict validation of each  
 3714 CIM-XML operation request. A validation-level of `loosely-validating` indicates that the WBEM  
 3715 server applies [loose validation](#) of each CIM-XML operation request.

3716 In the absence of this header, a WBEM client should assume that the WBEM server operates in strict  
 3717 validation mode.

## 3718 7.6 Other HTTP Methods

3719 This document does not in any way define or constrain the way a WBEM client, WBEM server, or WBEM  
 3720 listener uses any HTTP method other than those explicitly cited.

## 3721 7.7 Discovery and Addressing

3722 The target URI of the [CIM-XML operation request](#) is defined as the location of the [WBEM server](#). This  
 3723 document does not constrain the format of this URI other than it should be a valid URI ([RFC2396](#)) for  
 3724 describing an HTTP-addressable resource.

3725 An HTTP server that supports the CIM mapping defined in this document, and which supports the  
 3726 OPTIONS method, should include the following CIM extension header in an OPTIONS response:

- 3727 • CIMOM

3728 This header is defined as follows:

```

3729     CIMOM           = "CIMOM" ":" (absoluteURI | relativeURI)

```

3730 The terms `absoluteURI` and `relativeURI` are taken from [RFC2616](#); they indicate the location of the  
 3731 WBEM server for this HTTP server.

3732 If the CIMOM extension header is included in the response, the WBEM server shall declare it an optional  
 3733 extension header as described in 7.5.

3734 A [WBEM client](#) that needs to communicate with a WBEM server on an HTTP server should try an  
 3735 OPTIONS request to that HTTP server. If the OPTIONS request fails or the response does not include the  
 3736 CIM-CIMOM extension header, the WBEM client may assume that the value of CIM-CIMOM is the  
 3737 relative URI `cimom`.

3738 The DMTF recommends the use of the following well-known IP ports in compliant WBEM servers. This is  
 3739 a recommendation and not a requirement. The DMTF has registered these port addresses with IANA, so  
 3740 they are for the exclusive use of the DMTF.

- 3741 • CIM-XML (HTTP) 5988/tcp
- 3742 • CIM-XML (HTTP) 5988/udp
- 3743 • CIM-XML (HTTPS) 5989/tcp
- 3744 • CIM-XML (HTTPS) 5989/udp

3745 Other discovery mechanisms are outside the scope of this version of the specification.

3746 EXAMPLE 1:

```

3747     This example shows an HTTP server located at http://www.dmtf.org/ issuing an OPTIONS response
3748     to an HTTP client to indicate that its WBEM server is located at http://www.dmtf.org/access/cimom.

```

3749 HTTP/1.1 200 OK  
3750 Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=48  
3751 48-CIMOM: /access/cimom  
3752 ...

3753 EXAMPLE 2:

3754 If an HTTP server located at http://www.dmtf.org/ responds with a "501 Not Implemented" to an  
3755 OPTIONS request from a WBEM client, the WBEM client may then try to contact the WBEM server  
3756 at http://www.dmtf.org/cimom.

## 3757 7.8 Internationalization Considerations

3758 This clause defines the capabilities of the CIM HTTP mapping with respect to IETF policy guidelines on  
3759 character sets and languages ([RFC2277](#)).

3760 In this document, human-readable fields are contained within a response or request entity body. In all  
3761 cases, a human-readable content is encoded using XML (which explicitly provides for character set  
3762 tagging and encoding) and requires that XML processors read XML elements encoded, at minimum,  
3763 using the UTF-8 ([RFC2279](#)) encoding of the ISO 10646 multilingual plane.

3764 Properties that are not of type string or string array shall not be localized.

3765 Because keys are writeable only on instantiation, key values shall not be localized. See [DSP0004](#) for  
3766 details.

3767 XML examples in this document demonstrate the use of the charset parameter of the Content-Type  
3768 header, as defined in [RFC2616](#), as well as the XML attribute on the <?xml> processing instruction, which  
3769 together provide charset identification information for MIME and XML processors. This document  
3770 mandates that conforming applications shall support at least the "UTF-8" charset encoding ([RFC2277](#)) in  
3771 the Content-Type header and shall support the "UTF-8" value for the XML `encoding` attribute.

3772 XML also provides a language tagging capability for specifying the language of the contents of a  
3773 particular XML element, based on use of [IANA registered language tags \(RFC1766\)](#) in combination with  
3774 [ISO 639-1](#), in the `xml:lang` attribute of an XML element to identify the language of its content and  
3775 attributes. Section 3.10 of [RFC2616](#) defines how the two-character ISO 639-1 language code is used as  
3776 the primary-tag. The language-tag shall be registered by IANA.

3777 [DSP0201](#) declares this attribute on any XML elements. Therefore, conforming applications should use  
3778 this attribute when specifying the language in which a particular element is encoded for string and string  
3779 array attributes and qualifiers. See the usage [rules](#) on this element, which are defined by the World Wide  
3780 Web Consortium in [XML 1.0, second edition](#). The attribute may be scoped by the instance or a class and  
3781 should not be scoped by a property because instances or classes should be localized in one language.

3782 This document defines several names of HTTP headers and their values. These names are constructed  
3783 using standard encoding practices so that they always have an HTTP-safe ASCII representation.  
3784 Because these headers are not usually visible to users, they do not need to support encoding in multiple  
3785 character sets.

3786 [DSP0201](#) introduces several XML element names. Similarly, these names are not visible to an end user  
3787 and do not need to support multiple character set encodings.

3788 The [CIM model \(DSP0004\)](#) defines the subset of the Unicode character set that can be used to name  
3789 CIM elements (classes, instances, methods, properties, qualifiers, and method parameters). In general,  
3790 these characters appear as the value of XML attributes or as element content and are not displayed to  
3791 end users.

3792 Negotiation and notification of language settings is effected in this mapping using the standard [Accept-](#)  
3793 [Language](#) and [Content-Language](#) headers defined in [RFC1945](#) and [RFC2616](#).

## ANNEX A (Informative)

3794  
3795  
3796  
3797  
3798

### Examples of Message Exchanges

3799 This annex illustrates the protocol defined in this document with examples of valid HTTP  
3800 request/response exchanges. The examples are for illustration purposes only and are not considered part  
3801 of the specification.

3802 For clarity, additional white space is included in the examples, but such white space is not an intrinsic part  
3803 of such XML documents.

#### 3804 **A.1 Retrieval of a Single Class Definition**

3805 The following HTTP request illustrates how a client requests the class CIM\_VideoBIOSElement.

```

3806 M-POST /cimom HTTP/1.1
3807 HOST: http://www.myhost.com/
3808 Content-Type: application/xml; charset=utf-8
3809 Content-Length: xxxx
3810 Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3811 73-CIMOperation: MethodCall
3812 73-CIMMethod: GetClass
3813 73-CIMObject: root/cimv2
3814
3815 <?xml version="1.0" encoding="utf-8" ?>
3816 <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3817   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3818     <SIMPLEREQ>
3819       <IMETHODCALL NAME="GetClass">
3820         <LOCALNAMESPACEPATH>
3821           <NAMESPACE NAME="root"/>
3822           <NAMESPACE NAME="cimv2"/>
3823         </LOCALNAMESPACEPATH>
3824         <IPARAMVALUE NAME="ClassName">
3825           <CLASSNAME NAME="CIM_VideoBIOSElement"/>
3826         </IPARAMVALUE>
3827         <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
3828       </IMETHODCALL>
3829     </SIMPLEREQ>
3830   </MESSAGE>
3831 </CIM>

```

3832 Following is an HTTP response to the preceding request indicating success of the requested operation.  
3833 For clarity of exposition, the complete definition of the returned <CLASS> element is not shown.

```

3834 HTTP/1.1 200 OK
3835 Content-Type: application/xml; charset=utf-8
3836 Content-Length: xxxx
3837 Ext:
3838 Cache-Control: no-cache
3839 Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73

```

```

3840     73-CIMOperation: MethodResponse
3841
3842     <?xml version="1.0" encoding="utf-8" ?>
3843     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3844         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3845             <SIMPLERSP>
3846                 <IMETHODRESPONSE NAME="GetClass">
3847                     <IRETURNVALUE>
3848                         <CLASS NAME="CIM_VideoBIOSElement"
3849                             SUPERCLASS="CIM_SoftwareElement">
3850                             ...
3851                         </CLASS>
3852                     </IRETURNVALUE>
3853                 </IMETHODRESPONSE>
3854             </SIMPLERSP>
3855         </MESSAGE>
3856     </CIM>

```

## 3857 A.2 Retrieval of a Single Instance Definition

3858 The following HTTP request illustrates how a client requests the instance MyClass.MyKey="S3".

```

3859     M-POST /cimom HTTP/1.1
3860     HOST: http://www.myhost.com/
3861     Content-Type: application/xml; charset=utf-8
3862     Content-Length: xxxx
3863     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3864     73-CIMOperation: MethodCall
3865     73-CIMMethod: GetInstance
3866     73-CIMObject: root%2FmyNamespace
3867
3868     <?xml version="1.0" encoding="utf-8" ?>
3869     <CIM CIMVERSION="2.0" DTDVERSION="1.1">
3870         <MESSAGE ID="87855" PROTOCOLVERSION="1.0">
3871             <SIMPLEREQ>
3872                 <IMETHODCALL NAME="GetInstance">
3873                     <LOCALNAMESPACEPATH>
3874                         <NAMESPACE NAME="root"/>
3875                         <NAMESPACE NAME="myNamespace"/>
3876                     </LOCALNAMESPACEPATH>
3877                     <IPARAMVALUE NAME="InstanceName">
3878                         <INSTANCENAME CLASSNAME="MyClass">
3879                             <KEYBINDING NAME="MyKey"><KEYVALUE>S3</KEYVALUE></KEYBINDING>
3880                         </INSTANCENAME>
3881                     </IPARAMVALUE>
3882                     <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
3883                 </IMETHODCALL>
3884             </SIMPLEREQ>
3885         </MESSAGE>
3886     </CIM>

```

3887 Following is an HTTP response to the preceding request indicating an error because the specified  
3888 instance is not found.

```

3889 HTTP/1.1 200 OK
3890 Content-Type: application/xml; charset=utf-8
3891 Content-Length: xxxx
3892 Ext:
3893 Cache-Control: no-cache
3894 Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3895 73-CIMOperation: MethodResponse
3896
3897 <?xml version="1.0" encoding="utf-8" ?>
3898 <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3899   <MESSAGE ID="87885" PROTOCOLVERSION="1.0">
3900     <SIMPLERSP>
3901       <IMETHODRESPONSE NAME="GetInstance">
3902         <ERROR CODE="6" DESCRIPTION="Instance of MyClass not found"/>
3903       </IMETHODRESPONSE>
3904     </SIMPLERSP>
3905   </MESSAGE>
3906 </CIM>

```

### 3907 A.3 Deletion of a Single Class Definition

3908 The following HTTP request illustrates how a client deletes the class CIM\_VideoBIOSElement.

```

3909 M-POST /cimom HTTP/1.1
3910 HOST: http://www.myhost.com/
3911 Content-Type: application/xml; charset=utf-8
3912 Content-Length: xxxx
3913 Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3914 73-CIMOperation: MethodCall
3915 73-CIMMethod: DeleteClass
3916 73-CIMObject: root/cimv2
3917
3918 <?xml version="1.0" encoding="utf-8" ?>
3919 <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3920   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3921     <SIMPLEREQ>
3922       <IMETHODCALL NAME="DeleteClass">
3923         <LOCALNAMESPACEPATH>
3924           <NAMESPACE NAME="root"/>
3925           <NAMESPACE NAME="cimv2"/>
3926         </LOCALNAMESPACEPATH>
3927         <IPARAMVALUE NAME="ClassName">
3928           <CLASSNAME NAME="CIM_VideoBIOSElement"/>
3929         </IPARAMVALUE>
3930       </IMETHODCALL>
3931     </SIMPLEREQ>
3932   </MESSAGE>
3933 </CIM>

```

3934 Following is an HTTP response to the preceding request indicating failure of the preceding operation due  
3935 to the inability to delete instances of the class.

```

3936 HTTP/1.1 200 OK
3937 Content-Type: application/xml; charset=utf-8

```

```

3938     Content-Length: xxxx
3939     Ext:
3940     Cache-Control: no-cache
3941     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3942     73-CIMOperation: MethodResponse
3943
3944     <?xml version="1.0" encoding="utf-8" ?>
3945     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3946         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3947             <SIMPLERSP>
3948                 <IMETHODRESPONSE NAME="DeleteClass">
3949                     <ERROR CODE="9" DESCRIPTION="Class has non-deletable instances"/>
3950                 </IMETHODRESPONSE>
3951             </SIMPLERSP>
3952         </MESSAGE>
3953     </CIM>

```

#### 3954 **A.4 Deletion of a Single Instance Definition**

3955 The following HTTP request illustrates how a client deletes the instance MyClass.MyKey="S3".

```

3956     M-POST /cimom HTTP/1.1
3957     HOST: http://www.myhost.com/
3958     Content-Type: application/xml; charset=utf-8
3959     Content-Length: xxxx
3960     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3961     73-CIMOperation: MethodCall
3962     73-CIMMethod: DeleteInstance
3963     73-CIMObject: root%2FmyNamespace
3964
3965     <?xml version="1.0" encoding="utf-8" ?>
3966     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3967         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3968             <SIMPLEREQ>
3969                 <IMETHODCALL NAME="DeleteInstance">
3970                     <LOCALNAMESPACEPATH>
3971                         <NAMESPACE NAME="root"/>
3972                         <NAMESPACE NAME="myNamespace"/>
3973                     </LOCALNAMESPACEPATH>
3974                     <IPARAMVALUE NAME="InstanceName">
3975                         <INSTANCENAME CLASSNAME="MyClass">
3976                             <KEYBINDING NAME="MyKey">
3977                                 <KEYVALUE>S3</KEYVALUE>
3978                             </KEYBINDING>
3979                         </INSTANCENAME>
3980                     </IPARAMVALUE>
3981                 </IMETHODCALL>
3982             </SIMPLEREQ>
3983         </MESSAGE>
3984     </CIM>

```

3985 Following is an HTTP response to the preceding request indicating success of the preceding operation.

```

3986     HTTP/1.1 200 OK

```

```

3987     Content-Type: application/xml; charset=utf-8
3988     Content-Length: xxxx
3989     Ext:
3990     Cache-Control: no-cache
3991     Man: http://www.dmtf.org/cim/operation ; ns=73
3992     73-CIMOperation: MethodResponse
3993
3994     <?xml version="1.0" encoding="utf-8" ?>
3995     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3996         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3997             <SIMPLERSP>
3998                 <IMETHODRESPONSE NAME="DeleteInstance"/>
3999             </SIMPLERSP>
4000         </MESSAGE>
4001     </CIM>

```

## 4002 A.5 Creation of a Single Class Definition

4003 The following HTTP request illustrates how a client creates the class MySchema\_VideoBIOSElement as  
 4004 a subclass of CIM\_VideoBIOSElement. For clarity of exposition, most of the submitted <CLASS> element  
 4005 is omitted from the example.

```

4006     M-POST /cimom HTTP/1.1
4007     HOST: http://www.myhost.com/
4008     Content-Type: application/xml; charset=utf-8
4009     Content-Length: xxxx
4010     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4011     73-CIMOperation: MethodCall
4012     73-CIMMethod: CreateClass
4013     73-CIMObject: root/cimv2
4014
4015     <?xml version="1.0" encoding="utf-8" ?>
4016     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4017         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4018             <SIMPLEREQ>
4019                 <IMETHODCALL NAME="CreateClass">
4020                     <LOCALNAMESPACEPATH>
4021                         <NAMESPACE NAME="root"/>
4022                         <NAMESPACE NAME="cimv2"/>
4023                     </LOCALNAMESPACEPATH>
4024                     <IPARAMVALUE NAME="NewClass">
4025                         <CLASS NAME="MySchema_VideoBIOSElement"
4026                             SUPERCLASS="CIM_VideoBIOSElement">
4027                             ...
4028                         </CLASS>
4029                     </IPARAMVALUE>
4030                 </IMETHODCALL>
4031             </SIMPLEREQ>
4032         </MESSAGE>
4033     </CIM>

```

4034 Following is an HTTP response to the preceding request indicating success of the preceding operation.

```

4035     HTTP/1.1 200 OK

```

```

4036     Content-Type: application/xml; charset=utf-8
4037     Content-Length: xxxx
4038     Ext:
4039     Cache-Control: no-cache
4040     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4041     73-CIMOperation: MethodResponse
4042
4043     <?xml version="1.0" encoding="utf-8" ?>
4044     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4045         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4046             <SIMPLERSP>
4047                 <IMETHODRESPONSE NAME="CreateClass"/>
4048             </SIMPLERSP>
4049         </MESSAGE>
4050     </CIM>

```

## 4051 A.6 Creation of a Single Instance Definition

4052 The following HTTP request illustrates how a client creates an instance of the class  
4053 MySchema\_VideoBIOSElement. For clarity of exposition, most of the submitted <INSTANCE> element is  
4054 omitted from the example.

```

4055     M-POST /cimom HTTP/1.1
4056     HOST: http://www.myhost.com/
4057     Content-Type: application/xml; charset=utf-8
4058     Content-Length: xxxx
4059     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4060     73-CIMOperation: MethodCall
4061     73-CIMMethod: CreateInstance
4062     73-CIMObject: root/cimv2
4063
4064     <?xml version="1.0" encoding="utf-8" ?>
4065     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4066         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4067             <SIMPLEREQ>
4068                 <IMETHODCALL NAME="CreateInstance">
4069                     <LOCALNAMESPACEPATH>
4070                         <NAMESPACE NAME="root"/>
4071                         <NAMESPACE NAME="cimv2"/>
4072                     </LOCALNAMESPACEPATH>
4073                     <IPARAMVALUE NAME="NewInstance">
4074                         <INSTANCE CLASSNAME="CIM_VideoBIOSElement">
4075                             ...
4076                         </INSTANCE>
4077                     </IPARAMVALUE>
4078                 </IMETHODCALL>
4079             </SIMPLEREQ>
4080         </MESSAGE>
4081     </CIM>

```

4082 Following is an HTTP response to the preceding request indicating the success of the preceding  
4083 operation.

```

4084     HTTP/1.1 200 OK

```

```

4085     Content-Type: application/xml; charset=utf-8
4086     Content-Length: xxxx
4087     Ext:
4088     Cache-Control: no-cache
4089     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4090     73-CIMOperation: MethodResponse
4091
4092     <?xml version="1.0" encoding="utf-8" ?>
4093     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4094         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4095             <SIMPLERSP>
4096                 <IMETHODRESPONSE NAME="CreateInstance">
4097                     <IRETURNVALUE>
4098                         <INSTANCENAME CLASSNAME="MySchema_VideoBIOSElement">
4099                             <KEYBINDING NAME="Name"><KEYVALUE>S4</KEYVALUE></KEYBINDING>
4100                         </INSTANCENAME>
4101                     </IRETURNVALUE>
4102                 </IRETURNVALUE>
4103             </SIMPLERSP>
4104         </MESSAGE>
4105     </CIM>

```

## 4106 A.7 Enumeration of Class Names

4107 The following HTTP request illustrates how a client enumerates the names of all subclasses of the class  
 4108 CIM\_SoftwareElement.

```

4109     M-POST /cimom HTTP/1.1
4110     HOST: http://www.myhost.com/
4111     Content-Type: application/xml; charset=utf-8
4112     Content-Length: xxxx
4113     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4114     73-CIMOperation: MethodCall
4115     73-CIMMethod: EnumerateClassNames
4116     73-CIMObject: root/cimv2
4117
4118     <?xml version="1.0" encoding="utf-8" ?>
4119     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4120         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4121             <SIMPLEREQ>
4122                 <IMETHODCALL NAME="EnumerateClassNames">
4123                     <LOCALNAMESPACEPATH>
4124                         <NAMESPACE NAME="root"/>
4125                         <NAMESPACE NAME="cimv2"/>
4126                     </LOCALNAMESPACEPATH>
4127                     <IPARAMVALUE NAME="ClassName">
4128                         <CLASSNAME NAME="CIM_SoftwareElement"/>
4129                     </IPARAMVALUE>
4130                     <IPARAMVALUE NAME="DeepInheritance">
4131                         <VALUE>FALSE</VALUE>
4132                     </IPARAMVALUE>
4133                 </IMETHODCALL>
4134             </SIMPLEREQ>

```

```
4135     </MESSAGE>
4136     </CIM>
```

4137 Following is an HTTP response to the preceding request indicating the success of the preceding  
4138 operation and returning the names of the requested subclasses.

```
4139     HTTP/1.1 200 OK
4140     Content-Type: application/xml; charset=utf-8
4141     Content-Length: xxxx
4142     Ext:
4143     Cache-Control: no-cache
4144     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4145     73-CIMOperation: MethodResponse
4146
4147     <?xml version="1.0" encoding="utf-8" ?>
4148     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4149         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4150             <SIMPLERSP>
4151                 <IMETHODRESPONSE NAME="EnumerateClassNames">
4152                     <IRETURNVALUE>
4153                         <CLASSNAME NAME="CIM_BIOSElement"/>
4154                         <CLASSNAME NAME="CIM_VideoBOISElement"/>
4155                     </IRETURNVALUE>
4156                 </IMETHODRESPONSE>
4157             </SIMPLERSP>
4158         </MESSAGE>
4159     </CIM>
```

## 4160 A.8 Enumeration of Instances

4161 The following HTTP request illustrates how a client enumerates all instances of the class  
4162 CIM\_LogicalDisk. For clarity of exposition, most of the returned instances are omitted from the example.

```
4163     M-POST /cimom HTTP/1.1
4164     HOST: http://www.myhost.com/
4165     Content-Type: application/xml; charset=utf-8
4166     Content-Length: xxxx
4167     Man: http://www.dmtf.org/cim/operation ; ns=73
4168     73-CIMOperation: MethodCall
4169     73-CIMMethod: EnumerateInstances
4170     73-CIMObject: root/cimv2
4171
4172     <?xml version="1.0" encoding="utf-8" ?>
4173     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4174         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4175             <SIMPLEREQ>
4176                 <IMETHODCALL NAME="EnumerateInstances">
4177                     <LOCALNAMESPACEPATH>
4178                         <NAMESPACE NAME="root"/>
4179                         <NAMESPACE NAME="cimv2"/>
4180                     </LOCALNAMESPACEPATH>
4181                     <IPARAMVALUE NAME="ClassName">
4182                         <CLASSNAME NAME="CIM_LogicalDisk"/>
4183                     </IPARAMVALUE>
```

```

4184         <IPARAMVALUE NAME="LocalOnly"><VALUE>TRUE</VALUE></IPARAMVALUE>
4185         <IPARAMVALUE NAME="DeepInheritance"><VALUE>TRUE</VALUE></IPARAMVALUE>
4186     </IMETHODCALL>
4187 </SIMPLEREQ>
4188 </MESSAGE>
4189 </CIM>

```

4190 Following is an HTTP response to the preceding request indicating success of the preceding operation,  
 4191 returning the requested instances.

```

4192     HTTP/1.1 200 OK
4193     Content-Type: application/xml; charset=utf-8
4194     Content-Length: xxxx
4195     Ext:
4196     Cache-Control: no-cache
4197     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4198     73-CIMOperation: MethodResponse
4199
4200 <?xml version="1.0" encoding="utf-8" ?>
4201 <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4202     <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4203         <SIMPLERSP>
4204             <IMETHODRESPONSE NAME="EnumerateInstances">
4205                 <IRETURNVALUE>
4206                     <VALUE.NAMEDINSTANCE>
4207                         <INSTANCENAME CLASSNAME="Erewhon_LogicalDisk">
4208                             ...
4209                         </INSTANCENAME>
4210                         <INSTANCE CLASSNAME="Erewhon_LogicalDisk">
4211                             ...
4212                         </INSTANCE>
4213                     </VALUE.NAMEDINSTANCE>
4214                     ...
4215                     <VALUE.NAMEDINSTANCE>
4216                         <INSTANCENAME CLASSNAME="Foobar_LogicalDisk">
4217                             ...
4218                         </INSTANCENAME>
4219                         <INSTANCE CLASSNAME="Foobar_LogicalDisk">
4220                             ...
4221                         </INSTANCE>
4222                     </VALUE.NAMEINSTANCE>
4223                 </IRETURNVALUE>
4224             </IMETHODRESPONSE>
4225         </SIMPLERSP>
4226     </MESSAGE>
4227 </CIM>

```

## 4228 A.9 Retrieval of a Single Property

4229 The following HTTP request illustrates how a client retrieves the FreeSpace property from the instance  
 4230 MyDisk.DeviceID="C:". This example demonstrates how to use the GetInstance operation with a property  
 4231 list filter instead of the deprecated GetProperty operation.

```

4232     M-POST /cimom HTTP/1.1

```

```

4233     HOST: http://www.myhost.com/
4234     Content-Type: application/xml; charset=utf-8
4235     Content-Length: xxxx
4236     Man: http://www.dmtf.org/cim/operation ; ns=73
4237     73-CIMOperation: MethodCall
4238     73-CIMMethod: GetInstance
4239     73-CIMObject: root%2FmyNamespace
4240
4241     <?xml version="1.0" encoding="utf-8" ?>
4242     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4243         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4244             <SIMPLEREQ>
4245                 <IMETHODCALL NAME="GetInstance">
4246                     <LOCALNAMESPACEPATH>
4247                         <NAMESPACE NAME="root"/>
4248                         <NAMESPACE NAME="myNamespace"/>
4249                     </LOCALNAMESPACEPATH>
4250                     <IPARAMVALUE NAME="InstanceName">
4251                         <INSTANCENAME CLASSNAME="MyDisk">
4252                             <KEYBINDING NAME="DeviceID">
4253                                 <KEYVALUE>C:</KEYVALUE>
4254                             </KEYBINDING>
4255                         </INSTANCENAME>
4256                     </IPARAMVALUE>
4257                     <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
4258                     <IPARAMVALUE NAME="PropertyList">
4259                         <VALUE>FreeSpace</VALUE>
4260                     </IPARAMVALUE>
4261                 </IMETHODCALL>
4262             </SIMPLEREQ>
4263         </MESSAGE>
4264     </CIM>

```

4265 Following is an HTTP response to the preceding request indicating success of the preceding operation,  
 4266 returning the requested instance with the requested property value.

```

4267     HTTP/1.1 200 OK
4268     Content-Type: application/xml; charset=utf-8
4269     Content-Length: xxxx
4270     Ext:
4271     Cache-Control: no-cache
4272     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4273     73-CIMOperation: MethodResponse
4274
4275     <?xml version="1.0" encoding="utf-8" ?>
4276     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4277         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4278             <SIMPLERSP>
4279                 <IMETHODRESPONSE NAME="GetInstance">
4280                     <IRETURNVALUE>
4281                         <INSTANCE CLASSNAME="Erewhon_LogicalDisk">
4282                             <PROPERTY NAME="FreeSpace" TYPE="uint32">
4283                                 <VALUE>6752332</VALUE>
4284                             </PROPERTY>

```

```

4285         </INSTANCE>
4286         </IRETURNVALUE>
4287         </IMETHODRESPONSE>
4288         </SIMPLERSP>
4289         </MESSAGE>
4290     </CIM>

```

## 4291 A.10 Execution of an Extrinsic Method

4292 The following HTTP request illustrates how a client executes the SetPowerState method on the instance  
4293 MyDisk.DeviceID="C:".

```

4294     M-POST /cimom HTTP/1.1
4295     HOST: http://www.myhost.com/
4296     Content-Type: application/xml; charset=utf-8
4297     Content-Length: xxxx
4298     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4299     73-CIMOperation: MethodCall
4300     73-CIMMethod: SetPowerState
4301     73-CIMObject: root%2FmyNamespace%3A%22C%22
4302
4303     <?xml version="1.0" encoding="utf-8" ?>
4304     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4305         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4306             <SIMPLEREQ>
4307                 <METHODCALL NAME="SetPowerState">
4308                     <LOCALINSTANCEPATH>
4309                         <LOCALNAMESPACEPATH>
4310                             <NAMESPACE NAME="root"/>
4311                             <NAMESPACE NAME="myNamespace"/>
4312                         </LOCALNAMESPACEPATH>
4313                         <INSTANCENAME CLASSNAME="MyDisk">
4314                             <KEYBINDING NAME="Name"><KEYVALUE>C:</KEYVALUE></KEYBINDING>
4315                         </INSTANCENAME>
4316                     </LOCALINSTANCEPATH>
4317                     <PARAMVALUE NAME="PowerState"><VALUE>1</VALUE></PARAMVALUE>
4318                     <PARAMVALUE NAME="Time">
4319                         <VALUE>00000001132312.000000:000</VALUE>
4320                     </PARAMVALUE>
4321                 </METHODCALL>
4322             </SIMPLEREQ>
4323         </MESSAGE>
4324     </CIM>

```

4325 Following is an HTTP response to the preceding request indicating the success of the preceding  
4326 operation.

```

4327     HTTP/1.1 200 OK
4328     Content-Type: application/xml; charset=utf-8
4329     Content-Length: xxxx
4330     Ext:
4331     Cache-Control: no-cache
4332     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4333     73-CIMOperation: MethodResponse

```

```

4334
4335     <?xml version="1.0" encoding="utf-8" ?>
4336     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4337         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4338             <SIMPLERSP>
4339                 <METHODRESPONSE NAME="SetPowerState">
4340                     <RETURNVALUE>
4341                         <VALUE>0</VALUE>
4342                     </RETURNVALUE>
4343                 </METHODRESPONSE>
4344             </SIMPLERSP>
4345         </MESSAGE>
4346     </CIM>

```

## 4347 A.11 Indication Delivery Example

4348 The following HTTP request illustrates the format for sending an indication of type CIM\_AlertIndication to  
 4349 a WBEM listener.

```

4350     M-POST /cimlistener/browser HTTP/1.1
4351     HOST: http://www.acme.com/
4352     Content-Type: application/xml; charset=utf-8
4353     Content-Length: XXX
4354     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=40
4355     40-CIMExport: MethodRequest
4356     40-CIMExportMethod: ExportIndication
4357
4358     <?xml version="1.0" encoding="utf-8" ?>
4359     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4360         <MESSAGE ID="1007" PROTOCOLVERSION="1.0">
4361             <SIMPLEEXPREQ>
4362                 <EXPMETHODCALL NAME="ExportIndication">
4363                     <EXPPARAMVALUE NAME="NewIndication">
4364                         <INSTANCE CLASSNAME="CIM_AlertIndication" >
4365                             <PROPERTY NAME="Description" TYPE="string">
4366                                 <VALUE>Sample CIM_AlertIndication indication</VALUE>
4367                             </PROPERTY>
4368                             <PROPERTY NAME="AlertType" TYPE="uint16">
4369                                 <VALUE>1</VALUE>
4370                             </PROPERTY>
4371                             <PROPERTY NAME="PerceivedSeverity" TYPE="uint16">
4372                                 <VALUE>3</VALUE>
4373                             </PROPERTY>
4374                             <PROPERTY NAME="ProbableCause" TYPE="uint16">
4375                                 <VALUE>2</VALUE>
4376                             </PROPERTY>
4377                             <PROPERTY NAME="IndicationTime" TYPE="datetime">
4378                                 <VALUE>20010515104354.000000:000</VALUE>
4379                             </PROPERTY>
4380                         </INSTANCE>
4381                     </EXPPARAMVALUE>
4382                 </EXPMETHODCALL>
4383             </SIMPLEEXPREQ>

```

4384           </MESSAGE>  
 4385           </CIM>

4386   Following is an HTTP response to the preceding request indicating a successful receipt by the WBEM  
 4387   listener.

4388           HTTP/1.1 200 OK  
 4389           Content-Type: application/xml; charset=utf-8  
 4390           Content-Length: 267  
 4391           Ext:  
 4392           Cache-Control: no-cache  
 4393           Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=40  
 4394           40-CIMExport: MethodResponse  
 4395  
 4396           <?xml version="1.0" encoding="utf-8" ?>  
 4397           <CIM CIMVERSION="2.0" DTDVERSION="2.0">  
 4398            <MESSAGE ID="1007" PROTOCOLVERSION="1.0">  
 4399             <SIMPLEEXPRSP>  
 4400              <EXPMETHODRESPONSE NAME="ExportIndication">  
 4401                <IRETURNVALUE></IRETURNVALUE>  
 4402              </EXPMETHODRESPONSE>  
 4403             </SIMPLEEXPRSP>  
 4404            </MESSAGE>  
 4405           </CIM>

## 4406   A.12 Subscription Example

4407   A WBEM client application activates a subscription by creating an instance of the  
 4408   CIM\_IndicationSubscription class, which defines an association between a CIM\_IndicationFilter (a filter)  
 4409   instance and a CIM\_IndicationHandler (a handler) instance. The CIM\_IndicationFilter instance defines the  
 4410   filter criteria and data project list to describe the desired indication stream. The CIM\_IndicationHandler  
 4411   instance defines the desired indication encoding, destination location, and protocol for delivering the  
 4412   indication stream.

4413   The following HTTP request illustrates how a client creates an instance of the class CIM\_IndicationFilter.  
 4414   Note that the exact syntax of the WMI Query Language is still under review and is subject to change.

4415           Host: bryce  
 4416           Content-Type: application/xml; charset=utf-8  
 4417           Content-Length: XXXX  
 4418           Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=20  
 4419           20-CIMProtocolVersion: 1.0  
 4420           20-CIMOperation: MethodCall  
 4421           20-CIMMethod: CreateInstance  
 4422           20-CIMObject: root/cimv2  
 4423  
 4424           <?xml version="1.0" encoding="utf-8"?>  
 4425           <CIM CIMVERSION="2.0" DTDVERSION="2.0">  
 4426            <MESSAGE ID="53000" PROTOCOLVERSION="1.0">  
 4427             <SIMPLEREQ>  
 4428              <IMETHODCALL NAME="CreateInstance">  
 4429                <LOCALNAMESPACEPATH>  
 4430                  <NAMESPACE NAME="root"/>  
 4431                  <NAMESPACE NAME="cimv2"/>  
 4432                </LOCALNAMESPACEPATH>

```

4433         <IPARAMVALUE NAME="NewInstance">
4434             <INSTANCE CLASSNAME="CIM_IndicationFilter">
4435                 <PROPERTY NAME="SystemCreationClassName" TYPE="string">
4436                     <VALUE>CIM_UnitaryComputerSystem</VALUE>
4437                 </PROPERTY>
4438                 <PROPERTY NAME="SystemName" TYPE="string">
4439                     <VALUE>server001.acme.com</VALUE>
4440                 </PROPERTY>
4441                 <PROPERTY NAME="CreationClassName" TYPE="string">
4442                     <VALUE>CIM_IndicationFilter</VALUE>
4443                 </PROPERTY>
4444                 <PROPERTY NAME="Name" TYPE="string">
4445                     <VALUE>ACMESubscription12345</VALUE>
4446                 </PROPERTY>
4447                 <PROPERTY NAME="SourceNamespace" TYPE="string">
4448                     <VALUE>root/cimv2</VALUE>
4449                 </PROPERTY>
4450                 <PROPERTY NAME="Query" TYPE="string">
4451                     <VALUE>
4452                         SELECT Description, AlertType, PerceivedSeverity,
4453                             ProbableCause, IndicationTime
4454                         FROM CIM_AlertIndication
4455                         WHERE PerceivedSeverity = 3
4456                     </VALUE>
4457                 </PROPERTY>
4458                 <PROPERTY NAME="QueryLanguage" TYPE="string">
4459                     <VALUE>WQL</VALUE>
4460                 </PROPERTY>
4461             </INSTANCE>
4462         </IPARAMVALUE>
4463     </IMETHODCALL>
4464 </SIMPLEREQ>
4465 </MESSAGE>
4466 </CIM>

```

4467 Following is an HTTP response to the preceding request indicating success of the preceding operation.

```

4468 HTTP/1.1 200 OK
4469 Content-Type: application/xml; charset=utf-8
4470 Content-Length: XXX
4471 Ext:
4472 Cache-Control: no-cache
4473 Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=28
4474 28-CIMOperation: MethodResponse
4475
4476 <?xml version="1.0" encoding="utf-8" ?>
4477 <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4478     <MESSAGE ID="53000" PROTOCOLVERSION="1.0">
4479         <SIMPLERSP>
4480             <IMETHODRESPONSE NAME="CreateInstance">
4481                 <IRETURNVALUE>
4482                     <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
4483                         <KEYBINDING NAME="SystemCreationClassName">
4484                             <KEYVALUE VALUETYPE="string">

```

```

4485         CIM_UnitaryComputerSystem
4486     </KEYVALUE>
4487 </KEYBINDING>
4488 <KEYBINDING NAME="SystemName">
4489     <KEYVALUE VALUETYPE="string">
4490         server001.acme.com
4491     </KEYVALUE>
4492 </KEYBINDING>
4493 <KEYBINDING NAME="CreationClassName">
4494     <KEYVALUE VALUETYPE="string">
4495         CIM_IndicationFilter
4496     </KEYVALUE>
4497 </KEYBINDING>
4498 <KEYBINDING NAME="Name">
4499     <KEYVALUE VALUETYPE="string">
4500         ACMESubscription12345
4501     </KEYVALUE>
4502 </KEYBINDING>
4503 </INSTANCENAME>
4504 </IRETURNVALUE>
4505 </IMETHODRESPONSE>
4506 </SIMPLERSP>
4507 </MESSAGE>
4508 </CIM>

```

4509 The following HTTP request illustrates how a client creates an instance of the class  
4510 CIM\_IndicationHandlerCIMXML.

```

4511 M-POST /cimom HTTP/1.1
4512 Host: bryce
4513 Content-Type: application/xml; charset=utf-8
4514 Content-Length: XXX
4515 Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=20
4516 20-CIMProtocolVersion: 1.0
4517 20-CIMOperation: MethodCall
4518 20-CIMMethod: CreateInstance
4519 20-CIMObject: root/cimv2
4520
4521 <?xml version="1.0" encoding="utf-8"?>
4522 <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4523     <MESSAGE ID="54000" PROTOCOLVERSION="1.0">
4524         <SIMPLEREQ>
4525             <IMETHODCALL NAME="CreateInstance">
4526                 <LOCALNAMESPACEPATH>
4527                     <NAMESPACE NAME="root"/>
4528                     <NAMESPACE NAME="cimv2"/>
4529                 </LOCALNAMESPACEPATH>
4530                 <IPARAMVALUE NAME="NewInstance">
4531                     <INSTANCE CLASSNAME="CIM_IndicationHandlerCIMXML">
4532                         <PROPERTY NAME="SystemCreationClassName" TYPE="string">
4533                             <VALUE>CIM_UnitaryComputerSystem</VALUE>
4534                         </PROPERTY>
4535                         <PROPERTY NAME="SystemName" TYPE="string">
4536                             <VALUE>server001.acme.com</VALUE>

```

```

4537         </PROPERTY>
4538         <PROPERTY NAME="CreationClassName" TYPE="string">
4539             <VALUE>CIM_IndicationHandlerCIMXML</VALUE>
4540         </PROPERTY>
4541         <PROPERTY NAME="Name" TYPE="string">
4542             <VALUE>ACMESubscription12345</VALUE>
4543         </PROPERTY>
4544         <PROPERTY NAME="Owner" TYPE="string">
4545             <VALUE>ACMEAlertMonitoringConsole</VALUE>
4546         </PROPERTY>
4547         <PROPERTY NAME="Destination" TYPE="string">
4548             <VALUE>HTTP://www.acme.com/cimlistener/browser</VALUE>
4549         </PROPERTY>
4550     </INSTANCE>
4551 </IPARAMVALUE>
4552 </IMETHODCALL>
4553 </SIMPLEREQ>
4554 </MESSAGE>
4555 </CIM>

```

4556 Following is an HTTP response to the preceding request indicating the success of the preceding  
4557 operation.

```

4558 HTTP/1.1 200 OK
4559 Content-Type: application/xml; charset=utf-8
4560 Content-Length: XXX
4561 Ext:
4562 Cache-Control: no-cache
4563 Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=27
4564 27-CIMOperation: MethodResponse
4565
4566 <?xml version="1.0" encoding="utf-8" ?>
4567 <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4568     <MESSAGE ID="54000" PROTOCOLVERSION="1.0">
4569         <SIMPLERSP>
4570             <IMETHODRESPONSE NAME="CreateInstance">
4571                 <IRETURNVALUE>
4572                     <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
4573                         <KEYBINDING NAME="SystemCreationClassName">
4574                             <KEYVALUE VALUETYPE="string">
4575                                 CIM_UnitaryComputerSystem
4576                             </KEYVALUE>
4577                         </KEYBINDING>
4578                         <KEYBINDING NAME="SystemName">
4579                             <KEYVALUE VALUETYPE="string">
4580                                 server001.acme.com
4581                             </KEYVALUE>
4582                         </KEYBINDING>
4583                         <KEYBINDING NAME="CreationClassName">
4584                             <KEYVALUE VALUETYPE="string">
4585                                 CIM_IndicationHandlerCIMXML
4586                             </KEYVALUE>
4587                         </KEYBINDING>
4588                         <KEYBINDING NAME="Name">

```

```

4589         <KEYVALUE VALUETYPE="string">
4590             ACMESubscription12345
4591         </KEYVALUE>
4592     </KEYBINDING>
4593 </INSTANCENAME>
4594 </IRETURNVALUE>
4595 </IMETHODRESPONSE>
4596 </SIMPLERSP>
4597 </MESSAGE>
4598 </CIM>

```

4599 The following HTTP request illustrates how a client creates an instance of the class  
4600 CIM\_IndicationSubscription.

```

4601 M-POST /cimom HTTP/1.1
4602 Host: bryce
4603 Content-Type: application/xml; charset=utf-8
4604 Content-Length: XXXX
4605 Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=55
4606 55-CIMProtocolVersion: 1.0
4607 55-CIMOperation: MethodCall
4608 55-CIMMethod: CreateInstance
4609 55-CIMObject: root/cimv2
4610
4611 <?xml version="1.0" encoding="utf-8"?>
4612 <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4613     <MESSAGE ID="55000" PROTOCOLVERSION="1.0">
4614         <SIMPLEREQ>
4615             <IMETHODCALL NAME="CreateInstance">
4616                 <LOCALNAMESPACEPATH>
4617                     <NAMESPACE NAME="root"/>
4618                     <NAMESPACE NAME="cimv2"/>
4619                 </LOCALNAMESPACEPATH>
4620                 <IPARAMVALUE NAME="NewInstance">
4621                     <INSTANCE CLASSNAME="CIM_IndicationSubscription">
4622                         <PROPERTY.REFERENCE NAME="Filter"
4623                             REFERENCECLASS="CIM_IndicationFilter">
4624                             <VALUE.REFERENCE>
4625                                 <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
4626                                     <KEYBINDING NAME="SystemCreationClassName">
4627                                         <KEYVALUE VALUETYPE="string">
4628                                             CIM_UnitaryComputerSystem
4629                                         </KEYVALUE>
4630                                     </KEYBINDING>
4631                                     <KEYBINDING NAME="SystemName">
4632                                         <KEYVALUE VALUETYPE="string">
4633                                             server001.acme.com
4634                                         </KEYVALUE>
4635                                     </KEYBINDING>
4636                                     <KEYBINDING NAME="CreationClassName">
4637                                         <KEYVALUE VALUETYPE="string">
4638                                             CIM_IndicationFilter
4639                                         </KEYVALUE>
4640                                     </KEYBINDING>

```

```

4641         <KEYBINDING NAME="Name">
4642             <KEYVALUE VALUETYPE="string">
4643                 ACMESubscription12345
4644             </KEYVALUE>
4645         </KEYBINDING>
4646     </INSTANCENAME>
4647 </VALUE.REFERENCE>
4648 </PROPERTY.REFERENCE>
4649 <PROPERTY.REFERENCE NAME="Handler"
4650     REFERENCECLASS="CIM_IndicationHandler">
4651     <VALUE.REFERENCE>
4652         <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
4653             <KEYBINDING NAME="SystemCreationClassName">
4654                 <KEYVALUE VALUETYPE="string">
4655                     CIM_UnitaryComputerSystem
4656                 </KEYVALUE>
4657             </KEYBINDING>
4658             <KEYBINDING NAME="SystemName">
4659                 <KEYVALUE VALUETYPE="string">
4660                     server001.acme.com
4661                 </KEYVALUE>
4662             </KEYBINDING>
4663             <KEYBINDING NAME="CreationClassName">
4664                 <KEYVALUE VALUETYPE="string">
4665                     CIM_IndicationHandlerCIMXML
4666                 </KEYVALUE>
4667             </KEYBINDING>
4668             <KEYBINDING NAME="Name">
4669                 <KEYVALUE VALUETYPE="string">
4670                     ACMESubscription12345
4671                 </KEYVALUE>
4672             </KEYBINDING>
4673         </INSTANCENAME>
4674     </VALUE.REFERENCE>
4675 </PROPERTY.REFERENCE>
4676 </INSTANCE>
4677 </IPARAMVALUE>
4678 </IMETHODCALL>
4679 </SIMPLEREQ>
4680 </MESSAGE>
4681 </CIM>

```

4682 Following is an HTTP response to the preceding request indicating the success of the preceding  
4683 operation.

```

4684     HTTP/1.1 200 OK
4685     Content-Type: application/xml; charset=utf-8
4686     Content-Length: XXXX
4687     Ext:
4688     Cache-Control: no-cache
4689     Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=75
4690     75-CIMOperation: MethodResponse
4691
4692     <?xml version="1.0" encoding="utf-8" ?>

```

```

4693 <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4694 <MESSAGE ID="55000" PROTOCOLVERSION="1.0">
4695 <SIMPLERSP>
4696 <IMETHODRESPONSE NAME="CreateInstance">
4697 <IRETURNVALUE>
4698 <INSTANCENAME CLASSNAME="CIM_IndicationSubscription">
4699 <KEYBINDING NAME="Filter">
4700 <VALUE.REFERENCE>
4701 <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
4702 <KEYBINDING NAME="SystemCreationClassName">
4703 <KEYVALUE VALUETYPE="string">
4704 CIM_UnitaryComputerSystem
4705 </KEYVALUE>
4706 </KEYBINDING>
4707 <KEYBINDING NAME="SystemName">
4708 <KEYVALUE VALUETYPE="string">
4709 server001.acme.com
4710 </KEYVALUE>
4711 </KEYBINDING>
4712 <KEYBINDING NAME="CreationClassName">
4713 <KEYVALUE VALUETYPE="string">
4714 CIM_IndicationFilter
4715 </KEYVALUE>
4716 </KEYBINDING>
4717 <KEYBINDING NAME="Name">
4718 <KEYVALUE VALUETYPE="string">
4719 ACMESubscription12345
4720 </KEYVALUE>
4721 </KEYBINDING>
4722 </INSTANCENAME>
4723 </VALUE.REFERENCE>
4724 </KEYBINDING>
4725 <KEYBINDING NAME="Handler">
4726 <VALUE.REFERENCE>
4727 <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
4728 <KEYBINDING NAME="SystemCreationClassName">
4729 <KEYVALUE VALUETYPE="string">
4730 CIM_UnitaryComputerSystem
4731 </KEYVALUE>
4732 </KEYBINDING>
4733 <KEYBINDING NAME="SystemName">
4734 <KEYVALUE VALUETYPE="string">
4735 server001.acme.com
4736 </KEYVALUE>
4737 </KEYBINDING>
4738 <KEYBINDING NAME="CreationClassName">
4739 <KEYVALUE VALUETYPE="string">
4740 CIM_IndicationHandlerCIMXML
4741 </KEYVALUE>
4742 </KEYBINDING>
4743 <KEYBINDING NAME="Name">
4744 <KEYVALUE VALUETYPE="string">
4745 ACMESubscription12345

```

```

4746             </KEYVALUE>
4747             </KEYBINDING>
4748             </INSTANCENAME>
4749             </VALUE.REFERENCE>
4750             </KEYBINDING>
4751             </INSTANCENAME>
4752             </IRETURNVALUE>
4753             </IMETHODRESPONSE>
4754             </SIMPLERSP>
4755             </MESSAGE>
4756             </CIM>

```

### 4757 A.13 Multiple Operations Example

4758 The following HTTP request illustrates how a client performs multiple operations. This example batches a  
 4759 GetClass, an EnumerateInstanceNames, and an EnumerateInstance operation on  
 4760 CIM\_ObjectManagerAdapter.

```

4761     POST /CIMOM1 HTTP/1.1
4762     Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
4763     Content-Length: XXX
4764     Host: localhost:5988
4765     CIMOperation: MethodCall
4766     CIMProtocolVersion: 1.0
4767     Content-Type: application/xml; charset=utf-8
4768     CIMBatch: CIMBatch
4769     <?xml version="1.0" encoding="UTF-8"?>
4770
4771     <CIM DTDVERSION="2.0" CIMVERSION="2.0">
4772         <MESSAGE ID="2004:2:5:1:1:11:41:1" PROTOCOLVERSION="1.0">
4773             <MULTIREQ>
4774                 <SIMPLEREQ>
4775                     <IMETHODCALL NAME="GetClass">
4776                         <LOCALNAMESPACEPATH>
4777                             <NAMESPACE NAME="interop" />
4778                         </LOCALNAMESPACEPATH>
4779                         <IPARAMVALUE NAME="ClassName">
4780                             <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
4781                         </IPARAMVALUE>
4782                         <IPARAMVALUE NAME="LocalOnly">
4783                             <VALUE>FALSE</VALUE>
4784                         </IPARAMVALUE>
4785                         <IPARAMVALUE NAME="IncludeClassOrigin">
4786                             <VALUE>TRUE</VALUE>
4787                         </IPARAMVALUE>
4788                     </IMETHODCALL>
4789                 </SIMPLEREQ>
4790                 <SIMPLEREQ>
4791                     <IMETHODCALL NAME="Associators">
4792                         <LOCALNAMESPACEPATH>
4793                             <NAMESPACE NAME="interop" />
4794                         </LOCALNAMESPACEPATH>
4795                         <IPARAMVALUE NAME="ObjectName">
4796                             <CLASSNAME NAME="CIM_ObjectManagerAdapter" />

```

```

4797         </IPARAMVALUE>
4798         <IPARAMVALUE NAME="IncludeQualifiers">
4799             <VALUE>TRUE</VALUE>
4800         </IPARAMVALUE>
4801         <IPARAMVALUE NAME="IncludeClassOrigin">
4802             <VALUE>TRUE</VALUE>
4803         </IPARAMVALUE>
4804     </IMETHODCALL>
4805 </SIMPLEREQ>
4806 <SIMPLEREQ>
4807     <IMETHODCALL NAME="EnumerateInstanceNames">
4808         <LOCALNAMESPACEPATH>
4809             <NAMESPACE NAME="interop" />
4810         </LOCALNAMESPACEPATH>
4811         <IPARAMVALUE NAME="ClassName">
4812             <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
4813         </IPARAMVALUE>
4814     </IMETHODCALL>
4815 </SIMPLEREQ>
4816 <SIMPLEREQ>
4817     <IMETHODCALL NAME="EnumerateInstances">
4818         <LOCALNAMESPACEPATH>
4819             <NAMESPACE NAME="interop" />
4820         </LOCALNAMESPACEPATH>
4821         <IPARAMVALUE NAME="ClassName">
4822             <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
4823         </IPARAMVALUE>
4824         <IPARAMVALUE NAME="LocalOnly">
4825             <VALUE>FALSE</VALUE>
4826         </IPARAMVALUE>
4827     </IMETHODCALL>
4828 </SIMPLEREQ>
4829 </MULTIREQ>
4830 </MESSAGE>
4831 </CIM>

```

4832 Following is the HTTP response to the preceding request indicating the success of the preceding  
4833 operation.

```

4834 HTTP/1.1 200 OK
4835 CIMOperation: MethodResponse
4836 Content-Length: XXX
4837
4838 <?xml version="1.0" encoding="UTF-8"?>
4839 <CIM DTDVERSION="2.0" CIMVERSION="2.0">
4840     <MESSAGE ID="2004:2:5:1:1:11:41:1" PROTOCOLVERSION="1.0">
4841         <MULTIRSP>
4842             <SIMPLERSP>
4843                 <IMETHODRESPONSE NAME="GetClass">
4844                     <IRETURNVALUE>
4845                         <CLASS SUPERCLASS="CIM_WBEMService"
4846                             NAME="CIM_ObjectManagerAdapter">
4847                             ...
4848                     </CLASS>

```

```

4849         </IRETURNVALUE>
4850     </IMETHODRESPONSE>
4851 </SIMPLERSP>
4852 <SIMPLERSP>
4853     <IMETHODRESPONSE NAME="Associators">
4854         <IRETURNVALUE>
4855             <VALUE.OBJECTWITHPATH>
4856                 ...
4857             </VALUE.OBJECTWITHPATH>
4858             <VALUE.OBJECTWITHPATH>
4859                 ...
4860             </VALUE.OBJECTWITHPATH>
4861                 ...
4862         </IRETURNVALUE>
4863     </IMETHODRESPONSE>
4864 </SIMPLERSP>
4865 <SIMPLERSP>
4866     <IMETHODRESPONSE NAME="EnumerateInstanceNames">
4867         <IRETURNVALUE>
4868             <INSTANCENAME CLASSNAME="WBEMSolutions_ObjectManagerAdapter">
4869                 ...
4870             </INSTANCENAME>
4871             <INSTANCENAME CLASSNAME="WBEMSolutions_ObjectManagerAdapter">
4872                 ...
4873             </INSTANCENAME>
4874                 ...
4875         </IRETURNVALUE>
4876     </IMETHODRESPONSE>
4877 </SIMPLERSP>
4878 <SIMPLERSP>
4879     <IMETHODRESPONSE NAME="EnumerateInstances">
4880         <IRETURNVALUE>
4881             <VALUE.NAMEDINSTANCE>
4882                 ...
4883             </VALUE.NAMEDINSTANCE>
4884             <VALUE.NAMEDINSTANCE>
4885                 ...
4886             </VALUE.NAMEDINSTANCE>
4887                 ...
4888         </IRETURNVALUE>
4889     </IMETHODRESPONSE>
4890 </SIMPLERSP>
4891 </MULTIRSP>
4892 </MESSAGE>
4893 </CIM>

```

## ANNEX B (informative)

4894  
4895  
4896  
4897  
4898

### LocalOnly Parameter Discussion

4899 This annex discusses the issues associated with the 1.1 definition of the `LocalOnly` parameter for the  
4900 `GetInstance` and `EnumerateInstances` operations.

#### 4901 **B.1 Explanation of the Deprecated 1.1 Interpretation**

4902 In April 2002, two DMTF Change Requests (CRs), CR809 (`EnumerateInstances`) and CR815  
4903 (`GetInstance`), were approved and incorporated into version 1.1 of this document to clarify the  
4904 interpretation of the `LocalOnly` flag for the `GetInstance` and `EnumerateInstances` operations. With these  
4905 CRs, the definition of the `LocalOnly` flag for these operations was modified to align with the  
4906 interpretation of this flag for the `GetClass` and `EnumerateClasses` operations. This change was incorrect,  
4907 resulted in reduced functionality, and introduced several backward compatibility issues.

4908 To clarify the difference between the 1.0 Interpretation and the 1.1 Interpretation (CR815), consider the  
4909 following example:

```
4910     class A {  
4911         [Key]  
4912         string name;  
4913         uint32 counter = 3;  
4914     };  
4915  
4916     class B : A {  
4917         uint32 moreData = 4;  
4918     };  
4919  
4920     instance of A {  
4921         name = "Roger";  
4922     };  
4923  
4924     instance of B {  
4925         name = "Karl";  
4926         counter = 3;  
4927         moreData = 5;  
4928     };  
4929  
4930     instance of B {  
4931         name = "Denise";  
4932         counter = 5;  
4933     };
```

4934 Assuming `PropertyList = NULL` and `LocalOnly = TRUE`, Table 7 shows the properties returned by  
 4935 a `GetInstance` operation.

4936 **Table 7 – Comparison of Properties Returned by `GetInstance` in Versions 1.0 and 1.1**

Instance	DSP0200 1.0 Interpretation	DSP0200 1.1 Interpretation
"Roger"	name	name, counter
"Karl"	name, counter, moreData	moreData
"Denise"	name, counter	moreData

4937 The properties returned using the 1.0 interpretation are consistent with the properties specified in the  
 4938 MOF instance definitions, and the properties returned using the 1.1 Interpretation are consistent with the  
 4939 properties defined in the class definitions.

4940 **B.2 Risks of Using the 1.1 Interpretation**

4941 The risks of using the 1.1 interpretation are as follows:

- 4942 1) Within the DMTF, promoting a property from a class to one of its superclasses is defined as a  
 4943 backward-compatible change that can be made in a minor revision of the CIM schema. With the 1.1  
 4944 interpretation, promoting a property to a superclass can cause backward-incompatible changes.

4945 Suppose, for example, version 1.0 of the schema includes the following definitions:

```

4946 class A {
4947     [Key]
4948     string name;
4949     uint32 counter = 3;
4950 };
4951
4952 class B : A {
4953     uint32 moreData = 4;
4954 };
    
```

4955 Now suppose that the schema is modified in version 1.1 to promote the property `moreData` from  
 4956 class B to class A.

```

4957 class A {
4958     [Key]
4959     string name;
4960     uint32 counter = 3;
4961     uint32 moreData = 4;
4962 };
4963
4964 class B : A {
4965 };
    
```

4966 Using these examples, Table 8 shows the properties returned by a call to `GetInstance` with  
 4967 `PropertyList = NULL` and `LocalOnly = TRUE`. With the 1.1 Interpretation, this schema  
 4968 change would affect the list of properties returned. When dealing with a WBEM server that complies  
 4969 with the 1.1 interpretation, applications must be designed to treat “promoting properties” as a  
 4970 backward-compatible change.

4971 **Table 8 – Comparison of Properties Returned by a Call to GetInstance in Versions 1.0 and 1.1**

Instance	Schema Version 1.0	Schema Version 1.1
of A	name, counter	name, counter, moreData
of B	moreData	none

4972

4973 2) The 1.1 Interpretation encourages application developers to use multiple operations to retrieve the  
 4974 properties of an instance. That is, a commonly-stated use model for the 1.1 interpretation is to  
 4975 selectively traverse subclasses getting additional properties of an instance. This practice significantly  
 4976 increases the risk that a client will construct an inconsistent instance. With both Interpretations,  
 4977 applications should be designed to ensure that dependent properties are retrieved together.

### 4978 **B.3 Techniques for Differentiating between the 1.0 Interpretation and 1.1** 4979 **Interpretation**

4980 For concrete classes, WBEM servers that comply with the 1.0 Interpretation return the value of all KEY  
 4981 properties not explicitly excluded by the `PropertyList` parameter. WBEM servers that comply with the  
 4982 1.1 interpretation return only the value of KEY properties explicitly defined in the class. Applications can  
 4983 use this difference to detect which interpretation is supported by a WBEM server.

**ANNEX C  
(normative)**

**Generic Operations Mapping**

4984  
4985  
4986  
4987  
4988

4989 This annex defines a mapping of generic operations (see [DSP0223](#)) to the CIM-XML protocol described  
4990 in this document.

4991 A main purpose of this mapping is to support the implementations of DMTF management profiles that  
4992 define operations in terms of generic operations, by providing them a translation from the generic  
4993 operation listed in the management profile, to the CIM-XML operation that actually needs to be  
4994 implemented.

**4995 C.1 Operations**

4996 This subclause defines for each generic operation, which CIM-XML operation needs to be supported in  
4997 order to support the respective generic operation.

4998 Table 9 lists the generic operations defined in [DSP0223](#) and for each of them, lists the name of the  
4999 corresponding CIM-XML operation and a link to the description subclause.

5000 **Table 9 – Mapping of generic operations to CIM-XML operations**

Generic Operation	CIM-XML Operation	Description
GetInstance	GetInstance	See C.1.1
DeleteInstance	DeleteInstance	See C.1.2
ModifyInstance	ModifyInstance	See C.1.3
CreateInstance	CreateInstance	See C.1.4
GetClassInstancesWithPath	EnumerateInstances	See C.1.5
GetClassInstancePaths	EnumerateInstanceNames	See C.1.6
GetAssociatedInstancesWithPath	Associators (ObjectName is an instance path)	See C.1.7
GetAssociatedInstancePaths	AssociatorNames (ObjectName is an instance path)	See C.1.8
GetReferencingInstancesWithPath	References (ObjectName is an instance path)	See C.1.9
GetReferencingInstancePaths	ReferenceNames (ObjectName is an instance path)	See C.1.10
OpenClassInstancesWithPath	OpenEnumerateInstances	See C.1.11
OpenClassInstancePaths	OpenEnumerateInstancePaths	See C.1.12
OpenAssociatedInstancesWithPath	OpenAssociatorInstances	See C.1.13
OpenAssociatedInstancePaths	OpenAssociatorInstanceNames	See C.1.14
OpenReferencingInstancesWithPath	OpenReferenceInstances	See C.1.15
OpenReferencingInstancePaths	OpenReferenceInstanceNames	See C.1.16
OpenQueryInstances	OpenQueryInstances	See C.1.17
PullInstancesWithPath	PullInstancesWithPath	See C.1.18
PullInstancePaths	PullInstancePaths	See C.1.19

Generic Operation	CIM-XML Operation	Description
PullInstances	PullInstances	See C.1.20
CloseEnumeration	CloseEnumeration	See C.1.21
EnumerationCount	EnumerationCount	See C.1.22
InvokeMethod	invocation of extrinsic non-static method	See C.1.23
InvokeStaticMethod	invocation of extrinsic static method	See C.1.24
GetClass	GetClass	See C.1.25
DeleteClass	DeleteClass	See C.1.26
ModifyClass	ModifyClass	See C.1.27
CreateClass	CreateClass	See C.1.28
GetTopClassesWithPath	EnumerateClasses (ClassName is NULL)	See C.1.29
GetTopClassPaths	EnumerateClassNames (ClassName is NULL)	See C.1.30
GetSubClassesWithPath	EnumerateClasses (ClassName is non-NULL)	See C.1.31
GetSubClassPaths	EnumerateClassNames (ClassName is non-NULL)	See C.1.32
GetAssociatedClassesWithPath	Associators (ObjectName is a class path)	See C.1.33
GetAssociatedClassPaths	AssociatorNames (ObjectName is a class path)	See C.1.34
GetReferencingClassesWithPath	References (ObjectName is a class path)	See C.1.35
GetReferencingClassPaths	ReferenceNames (ObjectName is a class path)	See C.1.36
GetQualifierType	GetQualifier	See C.1.37
DeleteQualifierType	DeleteQualifier	See C.1.38
ModifyQualifierType	SetQualifier (Qualifier exists)	See C.1.39
CreateQualifierType	SetQualifier (Qualifier does not exist)	See C.1.40
EnumerateQualifierTypesWithPath	EnumerateQualifiers	See C.1.41

5001 In the following subclauses, the CIM-XML Type listed in the tables is either an intrinsic CIM type (e.g.  
 5002 "boolean"), or one of the pseudo-types defined in this document (e.g. "instanceName").

5003 **C.1.1 GetInstance**

5004 **CIM-XML Operation Name:** GetInstance

5005 **Purpose:** Retrieve an instance given its instance path.

5006 **Operation Input Parameters:**

5007

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
N/A	N/A	IncludeQualifiers	boolean	See 2)
N/A	N/A	LocalOnly	boolean	See 3)

- 5008 1) The CIM-XML parameter *InstanceName* includes the model path portion of the instance path of the  
 5009 instance. The generic parameter *InstancePath* corresponds to the combination of the CIM-XML  
 5010 parameter *InstanceName* and the target namespace of the CIM-XML operation.
- 5011 2) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The  
 5012 defined behavior of generic operation *GetInstance* conforms to the behavior of CIM-XML operation  
 5013 *GetInstance* with *IncludeQualifiers=false*, which is the recommended value to be used for CIM-XML  
 5014 clients since version 1.2 of this document.
- 5015 3) The CIM-XML parameter *LocalOnly* has been deprecated in version 1.2 of this document. The  
 5016 defined behavior of generic operation *GetInstance* conforms to the behavior of CIM-XML operation  
 5017 *GetInstance* with *LocalOnly=false*, which is the recommended value to be used for CIM-XML clients  
 5018 since version 1.2 of this document.

5019 **Operation Output Parameters:**

5020

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
Instance	InstanceSpecification	return value	instance	

5021 **Optional behavior:**

- 5022 • CIM-XML allows implementations to optimize by not including properties in the returned  
 5023 instance that have a value of NULL.

5024 **Deviations:** None

5025 **C.1.2 DeleteInstance**

5026 **CIM-XML Operation Name:** DeleteInstance

5027 **Purpose:** Delete an instance given its instance path.

5028 **Operation Input Parameters:**

5029

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)

- 5030 1) The CIM-XML parameter *InstanceName* includes the model path portion of the instance path of the  
 5031 instance. The generic parameter *InstancePath* corresponds to the combination of the CIM-XML  
 5032 parameter *InstanceName* and the target namespace of the CIM-XML operation.

5033 **Operation Output Parameters:** None

5034 **Deviations:** None

5035 **C.1.3 ModifyInstance**

5036 **CIM-XML Operation Name:** ModifyInstance

5037 **Purpose:** Modify property values of an instance given its instance path.

5038 **Operation Input Parameters:**

5039

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	target namespace	N/A	See 1)
		ModifiedInstance	namedInstance	See 1)
ModifiedInstance	InstanceSpecification	ModifiedInstance	namedInstance	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
N/A	N/A	IncludeQualifiers	boolean	See 2)

5040 1) The CIM-XML parameter *ModifiedInstance* includes the model path portion of the instance path of  
 5041 the instance that is being modified, and the modified property values. The combination of the model  
 5042 path portion of the CIM-XML parameter *ModifiedInstance* and the target namespace of the CIM-XML  
 5043 operation corresponds to the generic parameter *InstancePath*.

5044 2) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The  
 5045 defined behavior of generic operation *ModifyInstance* conforms to the behavior of CIM-XML  
 5046 operation *ModifyInstance* with *IncludeQualifiers=false*, which is the recommended behavior for CIM-  
 5047 XML servers since version 1.2 of this document.

5048 **Operation Output Parameters:** None

5049 **Optional behavior:**

- 5050 • [DSP0223](#) permits conformant WBEM protocols to require that all properties exposed by the  
 5051 creation class of the instance referenced by *InstancePath* are supplied by the WBEM client with  
 5052 their modified values. CIM-XML does not require that, i.e. CIM-XML permits clients to supply  
 5053 modified values only for a subset of these properties and those not supplied are meant to be left  
 5054 unchanged by the operation.

5055 **Deviations:** None

5056 **C.1.4 CreateInstance**

5057 **CIM-XML Operation Name:** CreateInstance

5058 **Purpose:** Create a CIM instance given the class path of its creation class.

5059 **Operation Input Parameters:**

5060

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		NewInstance	instance	See 1)
NewInstance	InstanceSpecification	NewInstance	instance	

- 5061 1) The generic parameter *ClassPath* corresponds to the combination of the class name specified in the  
 5062 CIM-XML parameter *NewInstance* and the target namespace of the CIM-XML operation.

5063 **Operation Output Parameters:**

5064

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	return value	instanceName	

5065 **Optional behavior:** None

5066 **Deviations:** None

5067 **C.1.5 GetClassInstancesWithPath**

5068 **CIM-XML Operation Name:** EnumerateInstances

5069 **Purpose:** Retrieve the instances of a given class (including instances of its subclasses). The retrieved  
 5070 instances include their instance paths.

5071 **Operation Input Parameters:**

5072

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	DeepInheritance	boolean	See 2)
N/A	N/A	IncludeQualifiers	boolean	See 3)
N/A	N/A	LocalOnly	boolean	See 4)

- 5073 1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter  
 5074 *ClassName* and the target namespace of the CIM-XML operation.
- 5075 2) The generic parameter *ExcludeSubclassProperties* corresponds to the negated CIM-XML parameter  
 5076 *DeepInheritance*.
- 5077 3) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The  
 5078 defined behavior of generic operation *GetClassInstancesWithPath* conforms to the behavior of CIM-  
 5079 XML operation *EnumerateInstances* with *IncludeQualifiers=false*, which is the recommended value  
 5080 to be used for CIM-XML clients since version 1.2 of this document.
- 5081 4) The CIM-XML parameter *LocalOnly* has been deprecated in version 1.2 of this document. The  
 5082 defined behavior of generic operation *GetClassInstancesWithPath* conforms to the behavior of CIM-  
 5083 XML operation *EnumerateInstances* with *LocalOnly=false*, which is the recommended value to be  
 5084 used for CIM-XML clients since version 1.2 of this document.

5085 **Operation Output Parameters:**

5086

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification- WithPath [ ]	return value	namedInstance [ ]	See 1)

5087 1) The CIM-XML return value includes the set of property values including the model paths, but without  
 5088 namespace paths. The generic parameter *InstanceList* needs to contain the instance paths in  
 5089 addition to the set of property values. A CIM client side mapping layer can construct the instance  
 5090 paths from the model paths and the CIM-XML target namespace.

5091 **Optional behavior:**

- 5092 • CIM-XML allows implementations to optimize by not including properties in the returned  
 5093 instances that have a value of NULL.

5094 **Deviations:** None

5095 **C.1.6 GetClassInstancePaths**

5096 **CIM-XML Operation Name:** EnumerateInstanceNames

5097 **Purpose:** Retrieve the instance paths of the instances of a given class (including instances of its  
 5098 subclasses).

5099 **Operation Input Parameters:**

5100

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)

5101 1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter  
 5102 *ClassName* and the target namespace of the CIM-XML operation.

5103 **Operation Output Parameters:**

5104

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instanceName [ ]	See 1)

5105 1) The CIM-XML return value includes the set of model paths, but without namespace paths. The  
 5106 generic parameter *InstancePathList* needs to contain the instance paths, including namespace  
 5107 paths. A CIM client side mapping layer can construct the instance paths from the model paths and  
 5108 the CIM-XML target namespace.

5109 **Optional behavior:** None

5110 **Deviations:** None

5111 **C.1.7 GetAssociatedInstancesWithPath**

5112 **CIM-XML Operation Name:** *Associators* with *ObjectName* being an instance path

5113 **Purpose:** Retrieve the instances that are associated with a given source instance. The retrieved  
 5114 instances include their instance paths.

5115 **Operation Input Parameters:**

5116

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	N/A	N/A	See 2)
N/A	N/A	IncludeQualifiers	boolean	See 3)

5117 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML  
 5118 parameter *ObjectName* and the target namespace of the CIM-XML operation.

5119 The generic operation *GetAssociatedInstancesWithPath* corresponds to the CIM-XML operation  
 5120 *Associators* when an instance path is passed in for its *ObjectName* parameter. Using the CIM-XML  
 5121 operation *Associators* with a class path for its *ObjectName* parameter is covered by the generic  
 5122 operation *GetAssociatedClassesWithPath* (see C.1.33).

5123 2) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-  
 5124 XML parameter. Since the defined behavior of the CIM-XML operation will result in including  
 5125 subclass properties, a mapping layer on the CIM client side can implement the behavior defined by  
 5126 the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that  
 5127 parameter has a value of true.

5128 3) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The  
 5129 defined behavior of generic operation *GetAssociatedInstancesWithPath* conforms to the behavior of  
 5130 CIM-XML operation *Associators* with *IncludeQualifiers=false*, which is the recommended value to be  
 5131 used for CIM-XML clients since version 1.2 of this document.

5132 **Operation Output Parameters:**

5133

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	objectWithPath [ ]	

5134 **Optional behavior:**

- 5135 • CIM-XML allows implementations to optimize by not including properties in the returned
- 5136 instances that have a value of NULL.

5137 **Deviations:** None

5138 **C.1.8 GetAssociatedInstancePaths**

5139 **CIM-XML Operation Name:** AssociatorNames with ObjectName being an instance path

5140 **Purpose:** Retrieve the instance paths of the instances that are associated with a given source instance.

5141 **Operation Input Parameters:**

5142

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	

5143 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML

5144 parameter *ObjectName* and the target namespace of the CIM-XML operation.

5145 The generic operation *GetAssociatedInstancePaths* corresponds to the CIM-XML operation

5146 *AssociatorNames* when an instance path is passed in for its *ObjectName* parameter. Using the CIM-

5147 XML operation *AssociatorNames* with a class path for its *ObjectName* parameter is covered by the

5148 generic operation *GetAssociatedClassPaths* (see C.1.34).

5149 **Operation Output Parameters:**

5150

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	objectPath [ ]	

5151 **Optional behavior:** None

5152 **Deviations:** None

5153 **C.1.9 GetReferencingInstancesWithPath**

5154 **CIM-XML Operation Name:** References with ObjectName being an instance path

5155 **Purpose:** Retrieve the association instances that reference a given source instance. The retrieved

5156 instances include their instance paths.

5157 **Operation Input Parameters:**

5158

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See Error! Reference source not found.
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See Error! Reference source not found.
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	N/A	N/A	See 3)
N/A	N/A	IncludeQualifiers	boolean	See 4)

5159 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML  
5160 parameter *ObjectName* and the target namespace of the CIM-XML operation.

5161 The generic operation *GetReferencingInstancesWithPath* corresponds to the CIM-XML operation  
5162 *References* when an instance path is passed in for its *ObjectName* parameter. Using the CIM-XML  
5163 operation *References* with a class path for its *ObjectName* parameter is covered by the generic  
5164 operation *GetReferencingClassesWithPath* (see C.1.35).

5165 2) The CIM-XML operation *References* does not support a means to filter by class name or role name  
5166 of the associated classes on the other ends of the associations referencing the source instance. The  
5167 generic operation *GetReferencingInstancesWithPath* does support such filtering through its  
5168 parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the  
5169 CIM-XML operation will result in including association instances that these two parameters could  
5170 filter out, a mapping layer on the CIM client side can implement the behavior defined by these two  
5171 generic parameters by eliminating association instances if these filter parameters are used.

5172 3) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-  
5173 XML parameter. Since the defined behavior of the CIM-XML operation will result in including  
5174 subclass properties, a mapping layer on the CIM client side can implement the behavior defined by  
5175 the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that  
5176 parameter has a value of true.

5177 4) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The  
5178 defined behavior of generic operation *GetReferencingInstancesWithPath* conforms to the behavior of  
5179 CIM-XML operation *References* with *IncludeQualifiers=false*, which is the recommended value to be  
5180 used for CIM-XML clients since in version 1.2 of this document.

5181 **Operation Output Parameters:**

5182

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	objectWithPath [ ]	

5183 **Optional behavior:**

- 5184 • CIM-XML allows implementations to optimize by not including properties in the returned  
5185 instances that have a value of NULL.

5186 **Deviations:** None

5187 **C.1.10 GetReferencingInstancePaths**

5188 **CIM-XML Operation Name:** ReferenceNames with ObjectName being an instance path

5189 **Purpose:** Retrieve the instance paths of the association instances that reference a given source  
5190 instance.

5191 **Operation Input Parameters:**

5192

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See <b>Error! Reference source not found.</b> 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See 2)
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)

5193 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML  
5194 parameter *ObjectName* and the target namespace of the CIM-XML operation.

5195 The generic operation *GetReferencingInstancePaths* corresponds to the CIM-XML operation  
5196 *ReferenceNames* when an instance path is passed in for its *ObjectName* parameter. Using the CIM-  
5197 XML operation *ReferenceNames* with a class path for its *ObjectName* parameter is covered by the  
5198 generic operation *GetReferencingClassPaths* (see C.1.36).

5199 2) The CIM-XML operation *References* does not support a means to filter by class name or role name  
5200 of the associated classes on the other ends of the associations referencing the source instance. The  
5201 generic operation *GetReferencingInstancesWithPath* does support such filtering through its  
5202 parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the  
5203 CIM-XML operation will result in including association instances that these two parameters could  
5204 filter out, a mapping layer on the CIM client side can implement the behavior defined by these two  
5205 generic parameters by eliminating association instances if these filter parameters are used.

5206 **Operation Output Parameters:**

5207

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	objectPath [ ]	

5208 **Optional behavior:** None

5209 **Deviations:** None

5210 **C.1.11 OpenClassInstancesWithPath**

5211 **CIM-XML Operation Name:** OpenEnumerateInstances

5212 **Purpose:** Open an enumeration session for retrieving the instances of a class (including instances of its subclasses), and optionally retrieve a first set of those instances. The retrieved instances include their instance paths.

5215 **Operation Input Parameters:**

5216

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	N/A	N/A	See Error! Reference source not found.2)
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

- 5217 1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter
- 5218 *ClassName* and the target namespace of the CIM-XML operation.
- 5219 2) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-
- 5220 XML parameter. Since the defined behavior of the CIM-XML operation will result in including
- 5221 subclass properties, a mapping layer on the CIM client side can implement the behavior defined by
- 5222 the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that
- 5223 parameter has a value of true.

5224 **Operation Output Parameters:**

5225

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	instanceWithPath [ ]	

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5226 **Optional behavior:**

- 5227     • CIM-XML allows implementations to optimize by not including properties in the returned  
5228       instances that have a value of NULL.

5229 **Deviations:** None

5230 **C.1.12 OpenClassInstancePaths**

5231 **CIM-XML Operation Name:** OpenEnumerateInstancePaths

5232 **Purpose:** Open an enumeration session for retrieving the instance paths of the instances of a class  
5233 (including instances of its subclasses), and optionally retrieve a first set of those instance paths.

5234 **Operation Input Parameters:**

5235

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

- 5236 1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter  
5237 *ClassName* and the target namespace of the CIM-XML operation.

5238 **Operation Output Parameters:**

5239

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5240 **Optional behavior:** None

5241 **Deviations:** None

5242 **C.1.13 OpenAssociatedInstancesWithPath**

5243 **CIM-XML Operation Name:** OpenAssociatorInstances

5244 **Purpose:** Open an enumeration session for retrieving the instances that are associated with a given  
 5245 source instance, and optionally retrieve a first set of those instances. The retrieved instances include their  
 5246 instance paths.

5247 **Operation Input Parameters:**

5248

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	N/A	N/A	See 2)
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

- 5249 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML  
 5250 parameter *InstanceName* and the target namespace of the CIM-XML operation.
- 5251 2) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-  
 5252 XML parameter. Since the defined behavior of the CIM-XML operation will result in including  
 5253 subclass properties, a mapping layer on the CIM client side can implement the behavior defined by  
 5254 the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that  
 5255 parameter has a value of true.

5256 **Operation Output Parameters:**

5257

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	instanceWithPath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5258 **Optional behavior:**

- 5259 • CIM-XML allows implementations to optimize by not including properties in the returned
- 5260 instances that have a value of NULL.

5261 **Deviations:** None

5262 **C.1.14 OpenAssociatedInstancePaths**

5263 **CIM-XML Operation Name:** OpenAssociatorInstancePaths

5264 **Purpose:** Open an enumeration session for retrieving the instance paths of instances that are associated  
 5265 with a given source instance, and optionally retrieve a first set of those instance paths.

5266 **Operation Input Parameters:**

5267

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

- 5268 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML  
 5269 parameter *InstanceName* and the target namespace of the CIM-XML operation.

5270 **Operation Output Parameters:**

5271

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5272 **Optional behavior:** None

5273 **Deviations:** None

5274 **C.1.15 OpenReferencingInstancesWithPath**

5275 **CIM-XML Operation Name:** OpenReferenceInstances

5276 **Purpose:** Open an enumeration session for retrieving the association instances that reference a given  
 5277 source instance, and optionally retrieve a first set of those instances. The retrieved instances include their  
 5278 instance paths.

5279 **Operation Input Parameters:**

5280

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See 2)
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	N/A	N/A	See 3)
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

- 5281 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML  
 5282 parameter *InstanceName* and the target namespace of the CIM-XML operation.
- 5283 2) The CIM-XML operation *OpenReferenceInstances* does not support a means to filter by class name  
 5284 or role name of the associated classes on the other ends of the associations referencing the source  
 5285 instance. The generic operation *OpenReferencingInstancesWithPath* does support such filtering  
 5286 through its parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior  
 5287 of the CIM-XML operation will result in including association instances that these two parameters  
 5288 could filter out, a mapping layer on the CIM client side can implement the behavior defined by these  
 5289 two generic parameters by eliminating association instances if these filter parameters are used.
- 5290 3) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-  
 5291 XML parameter. Since the defined behavior of the CIM-XML operation will result in including  
 5292 subclass properties, a mapping layer on the CIM client side can implement the behavior defined by  
 5293 the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that  
 5294 parameter has a value of true.

5295 **Operation Output Parameters:**

5296

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
--------------	--------------	--------------	--------------	-------------

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	instanceWithPath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5297 **Optional behavior:**

- 5298 • CIM-XML allows implementations to optimize by not including properties in the returned  
5299 instances that have a value of NULL.

5300 **Deviations:** None

5301 **C.1.16 OpenReferencingInstancePaths**

5302 **CIM-XML Operation Name:** OpenReferenceInstancePaths

5303 **Purpose:** Open an enumeration session for retrieving the instance paths of association instances that  
5304 reference a given source instance, and optionally retrieve a first set of those instance paths.

5305 **Operation Input Parameters:**

5306

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See 2)
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

5307 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML  
5308 parameter *InstanceName* and the target namespace of the CIM-XML operation.

5309 2) The CIM-XML operation *OpenReferenceInstancePaths* does not support a means to filter by class  
5310 name or role name of the associated classes on the other ends of the associations referencing the  
5311 source instance. The generic operation *OpenReferencingInstancePaths* does support such filtering  
5312 through its parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior  
5313 of the CIM-XML operation will result in including association instances that these two parameters  
5314 could filter out, a mapping layer on the CIM client side can implement the behavior defined by these  
5315 two generic parameters by eliminating association instances if these filter parameters are used.

5316 **Operation Output Parameters:**

5317

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5318 **Optional behavior:** None5319 **Deviations:** None5320 **C.1.17 OpenQueryInstances**5321 **CIM-XML Operation Name:** OpenQueryInstances

5322 **Purpose:** Open an enumeration session for retrieving the instances representing a query result, and  
 5323 optionally retrieve a first set of those instances. The retrieved instances are not addressable and thus do  
 5324 not include any instance paths.

5325 **Operation Input Parameters:**

5326

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
QueryString	QueryString	FilterQuery	string	
QueryLanguage	QueryLanguage	FilterQueryLanguage	string	
ReturnQueryResult- Class	boolean	ReturnQueryResult- Class	boolean	
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

5327 **Operation Output Parameters:**

5328

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification [ ]	return value	instance [ ]	
QueryResultClass		QueryResultClass	class	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5329 **Optional behavior:**

- 5330 • CIM-XML allows implementations to optimize by not including properties in the returned  
 5331 instances that have a value of NULL.

5332 **Deviations:** None

5333 **C.1.18 PullInstancesWithPath**

5334 **CIM-XML Operation Name:** PullInstancesWithPath

5335 **Purpose:** Retrieve the next set of instances from an open enumeration session. The retrieved instances  
 5336 include their instance paths.

5337 **Operation Input Parameters:**

5338

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
MaxObjectCount	uint32	MaxObjectCount	uint32	

5339 **Operation Output Parameters:**

5340

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification- WithPath [ ]	return value	instanceWithPath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5341 **Optional behavior:**

- 5342 • CIM-XML allows implementations to optimize by not including properties in the returned  
 5343 instances that have a value of NULL.

5344 **Deviations:** None

5345 **C.1.19 PullInstancePaths**

5346 **CIM-XML Operation Name:** PullInstancePaths

5347 **Purpose:** Retrieve the next set of instance paths from an open enumeration session.

5348 **Operation Input Parameters:**

5349

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
MaxObjectCount	uint32	MaxObjectCount	uint32	

5350 **Operation Output Parameters:**

5351

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5352 **Optional behavior:** None

5353 **Deviations:** None

5354 **C.1.20 PullInstances**

5355 **CIM-XML Operation Name:** PullInstances

5356 **Purpose:** Retrieve the next set of instances from an open enumeration session. The retrieved instances  
 5357 do not include any instance paths.

5358 **Operation Input Parameters:**

5359

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
MaxObjectCount	uint32	MaxObjectCount	uint32	

5360 **Operation Output Parameters:**

5361

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification [ ]	return value	instance [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5362 **Optional behavior:**

- 5363 • CIM-XML allows implementations to optimize by not including properties in the returned  
 5364 instances that have a value of NULL.

5365 **Deviations:** None

5366 **C.1.21 CloseEnumeration**

5367 **CIM-XML Operation Name:** CloseEnumeration

5368 **Purpose:** Close an open enumeration session.

5369 **Operation Input Parameters:**

5370

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	

5371 **Operation Output Parameters:** None

5372 **Optional behavior:** None

5373 **Deviations:** None

5374 **C.1.22 EnumerationCount**

5375 **CIM-XML Operation Name:** EnumerationCount

5376 **Purpose:** Estimate the total number of remaining items in an open enumeration session.

5377 **Operation Input Parameters:**

5378

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	

5379 **Operation Output Parameters:**

5380

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumerationCount	uint64	return value	uint64	

5381 **Optional behavior:** None

5382 **Deviations:** None

5383 **C.1.23 InvokeMethod**

5384 **CIM-XML Operation Name:** The generic operation *InvokeMethod* corresponds to CIM-XML extrinsic  
 5385 method invocation on an instance. CIM-XML extrinsic method invocation on a class is covered by the  
 5386 generic operation *InvokeStaticMethod* (see C.1.24).

5387 **Purpose:** Invoke a method on an instance.

5388 **Operation Input Parameters:**

5389 This document does not define an operation name or parameters for extrinsic method invocation.  
 5390 [DSP0201](#) defines the input and output parameters for extrinsic method invocation by means of the  
 5391 attributes and child elements of the XML elements METHODCALL and METHODRESPONSE. The table  
 5392 below therefore uses the names of these attributes and child elements in the mapping to generic  
 5393 operation parameters.

5394

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	target namespace	N/A	See 1)
		LOCALINSTANCE-PATH child element	N/A	See 1)
MethodName	MethodName	NAME attribute	N/A	
InParmValues	ParameterValue [ ]	set of PARAMVALUE child elements	N/A	

5395 1) The CIM-XML element *LOCALINSTANCEPATH* includes the model path portion of the instance path  
 5396 of the instance. The generic parameter *InstancePath* corresponds to the combination of the CIM-  
 5397 XML element *LOCALINSTANCEPATH* and the target namespace of the CIM-XML operation.

5398 **Operation Output Parameters:**

5399

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
OutParmValues	ParameterValue [ ]	set of PARAMVALUE child elements	N/A	
ReturnValue	ReturnValue	RETURNVALUE child element	N/A	

5400 **Optional behavior:** None

5401 **Deviations:** None

5402 **C.1.24 InvokeStaticMethod**

5403 **CIM-XML Operation Name:** The generic operation *InvokeStaticMethod* corresponds to CIM-XML  
 5404 extrinsic method invocation on a class. CIM-XML extrinsic method invocation on an instance is covered  
 5405 by the generic operation *InvokeMethod* (see C.1.23).

5406 **Purpose:** Invoke a static method on a class.

5407 **Operation Input Parameters:**

5408 This document does not define an operation name or parameters for extrinsic method invocation.  
 5409 [DSP0201](#) defines the input and output parameters for extrinsic method invocation by means of the  
 5410 attributes and child elements of the XML elements METHODCALL and METHODRESPONSE. The table  
 5411 below therefore uses the names of these attributes and child elements in the mapping to generic  
 5412 operation parameters.

5413

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		LOCALCLASSPATH child element	N/A	See 1)
MethodName	MethodName	NAME attribute	N/A	
InParmValues	ParameterValue [ ]	set of PARAMVALUE child elements	N/A	

- 5414 1) The CIM-XML element *LOCALCLASSPATH* includes the model path portion of the class path of the  
 5415 class. The generic parameter *ClassPath* corresponds to the combination of the CIM-XML element  
 5416 *LOCALCLASSPATH* and the target namespace of the CIM-XML operation.

5417 **Operation Output Parameters:**

5418

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
OutParmValues	ParameterValue [ ]	set of PARAMVALUE child elements	N/A	
ReturnValue	ReturnValue	RETURNVALUE child element	N/A	

5419 **Optional behavior:** None

5420 **Deviations:** None

5421 **C.1.25 GetClass**

5422 **CIM-XML Operation Name:** GetClass

5423 **Purpose:** Retrieve a class given its class path.

5424 **Operation Input Parameters:**

5425

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
N/A	N/A	LocalOnly	boolean	See 2)

- 5426 1) The CIM-XML parameter *ClassName* specifies the class name. The generic parameter *ClassPath*  
 5427 corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of  
 5428 the CIM-XML operation.
- 5429 2) The defined behavior of generic operation *GetClass* conforms to the behavior of CIM-XML operation  
 5430 *GetClass* with *LocalOnly=false*.

5431 **Operation Output Parameters:**

5432

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
Class	ClassSpecification-WithPath	return value	class	See 1)

- 5433 1) The CIM-XML return value includes the class declaration, without any class path information. The  
 5434 generic parameter *Class* needs to contain the class path in addition to the class declaration. A CIM  
 5435 client side mapping layer can remember the class path provided in the generic input parameter  
 5436 *ClassPath*, and add that to the generic output parameter *Class*.

5437 **Optional behavior:** None

5438 **Deviations:** None

5439 **C.1.26 DeleteClass**

5440 **CIM-XML Operation Name:** DeleteClass

5441 **Purpose:** Delete a class given its class path.

5442 **Operation Input Parameters:**

5443

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)
DeleteDependents	boolean	N/A	N/A	See 2)

- 5444 1) The CIM-XML parameter *ClassName* specifies the class name. The generic parameter *ClassPath*  
 5445 corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of  
 5446 the CIM-XML operation.
- 5447 2) **EXPERIMENTAL:** The experimental generic parameter *DeleteDependents* indicates whether  
 5448 dependent classes and instances are to be deleted as well. [DSP0223](#) defines the generic parameter  
 5449 *DeleteDependents* as optional. CIM-XML does not support deleting dependent classes and  
 5450 instances.

5451 **Operation Output Parameters:** None

5452 **Deviations:** None

5453 **C.1.27 ModifyClass**

5454 **CIM-XML Operation Name:** ModifyClass

5455 **Purpose:** Modify a class given its class path.

5456 **Operation Input Parameters:**

5457

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ModifiedClass	class	See 1)
ModifiedClass	ClassSpecification	ModifiedClass	class	

5458 1) The CIM-XML parameter *ModifiedClass* includes the name of the class that is being modified, and  
 5459 the modified class declaration. The combination of the class name portion of the CIM-XML  
 5460 parameter *ModifiedClass* and the target namespace of the CIM-XML operation corresponds to the  
 5461 generic parameter *ClassPath*.

5462 **Operation Output Parameters:** None

5463 **Optional behavior:** None

5464 **Deviations:** None

5465 **C.1.28 CreateClass**

5466 **CIM-XML Operation Name:** CreateClass

5467 **Purpose:** Create a class.

5468 **Operation Input Parameters:**

5469

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
NewClass	ClassSpecification	NewClass	class	

5470 **Operation Output Parameters:** None

5471 **Optional behavior:** None

5472 **Deviations:** None

5473 **C.1.29 GetTopClassesWithPath**

5474 **CIM-XML Operation Name:** EnumerateClasses with ClassName being NULL

5475 **Purpose:** Retrieve the top classes (i.e., classes that have no superclasses) of a given namespace. The  
 5476 retrieved classes include their class paths.

5477 **Operation Input Parameters:**

5478

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
IncludeSubclasses	boolean	DeepInheritance	boolean	

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
N/A	N/A	ClassName	className	See 1)
N/A	N/A	LocalOnly	boolean	See 2)

- 5479 1) The defined behavior of generic operation *GetTopClassesWithPath* conforms to the behavior of CIM-XML operation *EnumerateClasses* with *ClassName=NULL*.
- 5480
- 5481 2) The defined behavior of generic operation *GetTopClassesWithPath* conforms to the behavior of CIM-XML operation *EnumerateClasses* with *LocalOnly=false*.
- 5482

5483 **Operation Output Parameters:**

5484

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassList	ClassSpecification-WithPath [ ]	return value	class [ ]	See 1)

- 5485 1) The CIM-XML return value includes the set of class declarations including class names, but without a class path. The generic parameter *ClassList* needs to contain the class path in addition to the class declaration. A CIM client side mapping layer can construct the class paths from the class names and the CIM-XML target namespace.
- 5486
- 5487
- 5488

5489 **Optional behavior:** None

5490 **Deviations:** None

5491 **C.1.30 GetTopClassPaths**

5492 **CIM-XML Operation Name:** *EnumerateClassNames* with *ClassName* being NULL

5493 **Purpose:** Retrieve the class paths of the top classes (i.e., classes that have no superclasses) of a given namespace.

5494

5495 **Operation Input Parameters:**

5496

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
IncludeSubclasses	boolean	DeepInheritance	boolean	
N/A	N/A	ClassName	className	See 1)

- 5497 1) The defined behavior of generic operation *GetTopClassPaths* conforms to the behavior of CIM-XML operation *EnumerateClassNames* with *ClassName=NULL*.
- 5498

5499 **Operation Output Parameters:**

5500

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
--------------	--------------	--------------	--------------	-------------

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	className [ ]	See 1)

5501 1) The CIM-XML return value includes the set of class names, but without a class path. The generic  
 5502 parameter *ClassPathList* needs to contain the class paths. A CIM client side mapping layer can  
 5503 construct the class paths from the class names and the CIM-XML target namespace.

5504 **Optional behavior:** None

5505 **Deviations:** None

5506 **C.1.31 GetSubClassesWithPath**

5507 **CIM-XML Operation Name:** EnumerateClasses with ClassName being non-NULL

5508 **Purpose:** Retrieve the subclasses of a given class. The retrieved classes include their class paths.

5509 **Operation Input Parameters:**

5510

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1), 2)
IncludeSubclasses	boolean	DeepInheritance	boolean	
IncludeInherited-Elements	boolean	LocalOnly	boolean	See 3)
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	

5511 1) The CIM-XML parameter *ClassName* specifies the class name. The generic parameter *ClassPath*  
 5512 corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of  
 5513 the CIM-XML operation.

5514 2) The defined behavior of generic operation *GetSubClassesWithPath* conforms to the behavior of CIM-  
 5515 XML operation *EnumerateClasses* with *ClassName* being *non-NULL*.

5516 3) The generic parameter *IncludeInheritedElements* corresponds to the negated CIM-XML parameter  
 5517 *LocalOnly*.

5518 **Operation Output Parameters:**

5519

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassList	ClassSpecification- WithPath [ ]	return value	class [ ]	See 1)

5520 1) The CIM-XML return value includes the set of class declarations including class names, but without a  
 5521 class path. The generic parameter *ClassList* needs to contain the class path in addition to the class  
 5522 declaration. A CIM client side mapping layer can construct the class paths from the class names and  
 5523 the CIM-XML target namespace.

5524 **Optional behavior:** None

5525 **Deviations:** None

5526 **C.1.32 GetSubClassPaths**

5527 **CIM-XML Operation Name:** EnumerateClassNames with ClassName being non-NULL

5528 **Purpose:** Retrieve the class paths of the subclasses of a given class.

5529 **Operation Input Parameters:**

5530

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1), 2)
IncludeSubclasses	boolean	DeepInheritance	boolean	

5531 1) The CIM-XML parameter *ClassName* specifies the class name. The generic parameter *ClassPath*  
 5532 corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of  
 5533 the CIM-XML operation.

5534 2) The defined behavior of generic operation *GetSubClassPaths* conforms to the behavior of CIM-XML  
 5535 operation *EnumerateClassNames* with *ClassName* being *non-NULL*.

5536 **Operation Output Parameters:**

5537

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	className [ ]	See 1)

5538 1) The CIM-XML return value includes the set of class names, but without a class path. The generic  
 5539 parameter *ClassPathList* needs to contain the class paths. A CIM client side mapping layer can  
 5540 construct the class paths from the class names and the CIM-XML target namespace.

5541 **Optional behavior:** None

5542 **Deviations:** None

5543 **C.1.33 GetAssociatedClassesWithPath**

5544 **CIM-XML Operation Name:** Associators with ObjectName being a class path

5545 **Purpose:** Retrieve the classes that are associated with a given source class. The retrieved classes  
 5546 include their class paths.

5547 **Operation Input Parameters:**

5548

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
RoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	

5549 1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter  
 5550 *ObjectName* and the target namespace of the CIM-XML operation.

5551 The generic operation *GetAssociatedClassesWithPath* corresponds to the CIM-XML operation  
 5552 *Associators* when a class path is passed in for its *ObjectName* parameter. Using the CIM-XML  
 5553 operation *Associators* with an instance path for its *ObjectName* parameter is covered by the generic  
 5554 operation *GetAssociatedInstancesWithPath* (see C.1.7).

5555 **Operation Output Parameters:**

5556

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassList	ClassSpecification- WithPath [ ]	return value	objectWithPath [ ]	

5557 **Optional behavior:** None

5558 **Deviations:** None

5559 **C.1.34 GetAssociatedClassPaths**

5560 **CIM-XML Operation Name:** AssociatorNames with ObjectName being a class path

5561 **Purpose:** Retrieve the class paths of the classes that are associated with a given source class.

5562 **Operation Input Parameters:**

5563

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
RoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	

5564 1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter  
5565 *ObjectName* and the target namespace of the CIM-XML operation.

5566 The generic operation *GetAssociatedClassPaths* corresponds to the CIM-XML operation  
5567 *AssociatorNames* when a class path is passed in for its *ObjectName* parameter. Using the CIM-XML  
5568 operation *AssociatorNames* with an instance path for its *ObjectName* parameter is covered by the  
5569 generic operation *GetAssociatedInstancePaths* (see C.1.8).

5570 **Operation Output Parameters:**

5571

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	objectPath [ ]	

5572 **Optional behavior:** None

5573 **Deviations:** None

5574 **C.1.35 GetReferencingClassesWithPath**

5575 **CIM-XML Operation Name:** References with ObjectName being a class path

5576 **Purpose:** Retrieve the association classes that reference a given source class. The retrieved classes  
5577 include their class paths.

5578 **Operation Input Parameters:**

5579

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See 2)
RoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	

5580 1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter  
5581 *ObjectName* and the target namespace of the CIM-XML operation.

5582 The generic operation *GetReferencingClassesWithPath* corresponds to the CIM-XML operation  
5583 *References* when a class path is passed in for its *ObjectName* parameter. Using the CIM-XML  
5584 operation *References* with an instance path for its *ObjectName* parameter is covered by the generic  
5585 operation *GetReferencingInstancesWithPath* (see C.1.9).

5586 2) The CIM-XML operation *References* does not support a means to filter by class name or role name  
5587 of the associated classes on the other ends of the associations referencing the source class. The  
5588 generic operation *GetReferencingClassesWithPath* does support such filtering through its

5589 parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the  
 5590 CIM-XML operation will result in including association classes that these two parameters could filter  
 5591 out, a mapping layer on the CIM client side can implement the behavior defined by these two generic  
 5592 parameters by eliminating association classes if these filter parameters are used.

5593 **Operation Output Parameters:**

5594

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	objectWithPath [ ]	

5595 **Optional behavior:** None

5596 **Deviations:** None

5597 **C.1.36 GetReferencingClassPaths**

5598 **CIM-XML Operation Name:** ReferenceNames with ObjectName being a class path

5599 **Purpose:** Retrieve the class paths of the association classes that reference a given class.

5600 **Operation Input Parameters:**

5601

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See 2)
RoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)

5602 1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter  
 5603 *ObjectName* and the target namespace of the CIM-XML operation.

5604 The generic operation *GetReferencingClassPaths* corresponds to the CIM-XML operation  
 5605 *ReferenceNames* when a class path is passed in for its *ObjectName* parameter. Using the CIM-XML  
 5606 operation *ReferenceNames* with an instance path for its *ObjectName* parameter is covered by the  
 5607 generic operation *GetReferencingInstancePaths* (see C.1.10).

5608 2) The CIM-XML operation *References* does not support a means to filter by class name or role name  
 5609 of the associated classes on the other ends of the associations referencing the source class. The  
 5610 generic operation *GetReferencingClassesWithPath* does support such filtering through its  
 5611 parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the  
 5612 CIM-XML operation will result in including association classes that these two parameters could filter  
 5613 out, a mapping layer on the CIM client side can implement the behavior defined by these two generic  
 5614 parameters by eliminating association classes if these filter parameters are used.

5615 **Operation Output Parameters:**

5616

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	objectPath [ ]	

5617 **Optional behavior:** None

5618 **Deviations:** None

5619 **C.1.37 GetQualifierType**

5620 **CIM-XML Operation Name:** GetQualifier

5621 **Purpose:** Retrieve a qualifier type given its qualifier type path.

5622 **Operation Input Parameters:**

5623

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypePath	QualifierTypePath	target namespace	N/A	See 1), 1)
		QualifierName	string	See 1), 1)

5624 1) The CIM-XML parameter *QualifierName* specifies the name of the qualifier type. The generic  
 5625 parameter *QualifierTypePath* corresponds to the combination of the CIM-XML parameter  
 5626 *QualifierName* and the target namespace of the CIM-XML operation.

5627 **Operation Output Parameters:**

5628

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierType	QualifierType	return value	qualifierDecl	See 1)

5629 1) The CIM-XML return value includes the qualifier type declaration including the qualifier type name,  
 5630 but without the namespace path portion of the full qualifier type path. The generic parameter  
 5631 *QualifierType* needs to contain the full qualifier type path in addition to the qualifier type declaration.  
 5632 A CIM client side mapping layer can remember the qualifier type path provided in the generic input  
 5633 parameter *QualifierTypePath*, and add that to the generic output parameter *QualifierType*.

5634 **Optional behavior:** None

5635 **Deviations:** None

5636 **C.1.38 DeleteQualifierType**

5637 **CIM-XML Operation Name:** DeleteQualifier

5638 **Purpose:** Delete a qualifier type given its qualifier type path.

5639 **Operation Input Parameters:**

5640

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypePath	QualifierTypePath	target namespace	N/A	See 1)
		QualifierName	string	See 1)

5641 1) The CIM-XML parameter *QualifierName* specifies the name of the qualifier type, i.e. the model path  
 5642 portion of its qualifier type path. The generic parameter *QualifierTypePath* corresponds to the  
 5643 combination of the CIM-XML parameter *QualifierName* and the target namespace of the CIM-XML  
 5644 operation.

5645 **Operation Output Parameters:** None

5646 **Deviations:** None

5647 **C.1.39 ModifyQualifierType**

5648 **CIM-XML Operation Name:** SetQualifier

5649 **Purpose:** Modify a qualifier type given its qualifier type path.

5650 **Operation Input Parameters:**

5651

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypePath	QualifierTypePath	target namespace	N/A	See 1)
		QualifierDeclaration	qualifierDecl	See 1)
ModifiedQualifierType	QualifierType	QualifierDeclaration	qualifierDecl	

5652 1) The CIM-XML parameter *QualifierDeclaration* includes the name of the qualifier type that is modified,  
 5653 i.e. the model path portion of its qualifier type, and the modified qualifier type declaration. The  
 5654 combination of the name of the qualifier type within the CIM-XML parameter *QualifierDeclaration* and  
 5655 the target namespace of the CIM-XML operation corresponds to the generic parameter  
 5656 *QualifierTypePath*.

5657 **Operation Output Parameters:** None

5658 **Optional behavior:** None

5659 **Deviations:**

- 5660 • The generic operation *ModifyQualifierType* is required to fail if invoked on a non-existing  
 5661 qualifier type. The CIM-XML operation *SetQualifier* creates the qualifier type in this case. This  
 5662 deviation covers only an error case. A CIM client side mapping layer can expose the generic  
 5663 operation behavior by first testing for the existence of the qualifier type using the CIM-XML  
 5664 operation *GetQualifier*, before modifying it.

5665 **C.1.40 CreateQualifierType**

5666 **CIM-XML Operation Name:** SetQualifier

5667 **Purpose:** Create a CIM qualifier type.

5668 **Operation Input Parameters:**

5669

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	See 1)
		QualifierDeclaration	qualifierDecl	See 1)
NewQualifierType	QualifierType	QualifierDeclaration	qualifierDecl	

5670 1) The generic parameter *NamespacePath* corresponds to the combination of the qualifier type name  
 5671 specified in the CIM-XML parameter *NewQualifierType* and the target namespace of the CIM-XML  
 5672 operation.

5673 **Operation Output Parameters:**

5674

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypePath	QualifierTypePath	return value	instanceName	

5675 **Optional behavior:** None

5676 **Deviations:**

- 5677
- 5678 • The generic operation *CreateQualifierType* is required to fail if invoked on an existing qualifier  
 5679 type. The CIM-XML operation *SetQualifier* modifies the qualifier type in this case. This deviation  
 5680 covers only an error case. A CIM client side mapping layer can expose the generic operation  
 5681 behavior by first testing for the existence of the qualifier type using the CIM-XML operation  
*GetQualifier*, before creating it.

5682 **C.1.41 EnumerateQualifierTypesWithPath**

5683 **CIM-XML Operation Name:** EnumerateQualifiers

5684 **Purpose:** Retrieve the qualifier types of a given namespace. The retrieved qualifier types include their  
 5685 qualifier type paths.

5686 **Operation Input Parameters:**

5687

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	

5688 **Operation Output Parameters:**

5689

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypeList	QualifierTypeWithPath [ ]	return value	qualifierDecl [ ]	See 1)

5690 1) The CIM-XML return value includes the set of qualifier type declarations including their names, but  
5691 without namespace paths. The generic parameter QualifierTypeList needs to contain the qualifier  
5692 type paths in addition to the set of qualifier type declarations. A CIM client side mapping layer can  
5693 construct the qualifier type paths from the qualifier names and the CIM-XML target namespace.

5694 **Optional behavior:** None

5695 **Deviations:** None

5696  
5697  
5698  
5699  
5700

## ANNEX D (informative)

### Change Log

Version	Date	Description
1.0	1999-06-02	DMTF Final Standard
1.1	2003-01-06	DMTF Final Standard
1.2	2007-01-09	DMTF Final Standard
1.3.0	2008-10-15	DMTF Final Standard
1.3.1	2009-07-29	DMTF Standard Release

Version	Date	Description
1.4.0a	2013-05-07	<p>Work in Progress, with the following changes:</p> <p>Changes:</p> <ul style="list-style-type: none"> <li>• Changed representation of enumeration context value from an ENUMERATIONCONTEXT element to a string using the VALUE element (see 5.4.2.24.2) (CRCIMXML00022.001)</li> <li>• Added requirement to support DMTF Filter Query Language (FQL) in pulled enumeration operations (see 5.4.2.24.2) (CRCIMXML00033.001)</li> <li>• Updated several normative references (see clause 2) (multiple CRs)</li> <li>• Lifted requirements in CreateInstance to initialize only with client-provided values, and in ModifyInstance to update only with client-provided values, to leave room for model-defined deviations (see 5.4.2.6 and 5.4.2.8). (CRCIMXML00036.000)</li> </ul> <p>Deprecations::</p> <ul style="list-style-type: none"> <li>• Deprecated use of CIM_ERR_INVALID_CLASS on ExportIndication operation (see 5.5.2.1) (CRCIMXML00021.000)</li> <li>• Deprecated the GetProperty and SetProperty operations (see 5.4.2.18 and 5.4.2.19) (CRCIMXML00027.000)</li> <li>• Deprecated the EnumerateInstances, EnumerateInstanceNames, ExecQuery, and the instance-level Associators, AssociatorNames, References and ReferenceNames operations (CRCIMXML00030.002)</li> </ul> <p>Additional Functions and Requirements:</p> <ul style="list-style-type: none"> <li>• Added support for operation correlators (see 5.3) (CRCIMXML00014.002)</li> </ul> <p>Clarifications:</p> <ul style="list-style-type: none"> <li>• Clarified HTTPS support (see 7.1) (CRCIMXML00010.004)</li> <li>• Clarified filter query in pulled enumerations (5.4.2.24.2) (CRCIMXML00019.001)</li> <li>• Added mapping to generic operations (see ANNEX C) (CRCIMXML00034.000)</li> </ul> <p>Editorial Changes:</p> <ul style="list-style-type: none"> <li>• Terminology cleanup (CRCIMXML00026.002)</li> </ul>

5701

## Bibliography

5702

5703 DMTF DSP0203, *DTD for Representation of CIM in XML 2.4*,  
5704 [http://www.dmtf.org/standards/published\\_documents/DSP0203\\_2.4.dtd](http://www.dmtf.org/standards/published_documents/DSP0203_2.4.dtd)

5705 DMTF DSP8044, *XSD for Representation of CIM in XML 2.4*,  
5706 [http://schemas.dmtf.org/wbem/wbem/cim-xml/2/dsp8044\\_2.4.xsd](http://schemas.dmtf.org/wbem/wbem/cim-xml/2/dsp8044_2.4.xsd)

5707 IETF RFC2068 (obsoleted by [RFC2616](#)), *Hypertext Transfer Protocol – HTTP/1.1*, January 1997,  
5708 <http://www.ietf.org/rfc/rfc2068.txt>

5709 IETF RFC2069 (obsoleted by [RFC2617](#)), *An Extension to HTTP: Digest Access Authentication*, January  
5710 1997,  
5711 <http://www.ietf.org/rfc/rfc2069.txt>

5712 ITU-T X.509: *Information technology - Open Systems Interconnection - The Directory: Public-key and*  
5713 *attribute certificate frameworks*,  
5714 <http://www.itu.int/rec/T-REC-X.509/en>

5715 SSL 2.0, *Hickman: The SSL Protocol, Draft 02*, Netscape Communications Corp., February 1995,  
5716 <http://www.mozilla.org/projects/security/pki/nss/ssl/draft02.html>

5717 SSL 3.0, *Freier, Karlton, and Kocher: The SSL Protocol, Version 3.0, Final Draft*, Netscape  
5718 *Communications Corp.*, November 1996,  
5719 <http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt>