



1  
2  
3  
4

**Document Identifier: DSP0223**

**Date: 2015-03-06**

**Version: 2.0.0**

## 5 **Generic Operations**

6 **Supersedes: 1.1**

7 **Document Type: Specification**

8 **Document Class: Normative**

9 **Document Status: Published**

10 **Document Language: en-US**

11

12 Copyright notice

13 Copyright © 2007–2015 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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28 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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

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

## Contents

35	Foreword .....	6
36	Acknowledgements .....	6
37	Document conventions .....	6
38	Typographical conventions .....	6
39	Experimental material .....	6
40	1 Scope .....	7
41	2 Normative references .....	7
42	3 Terms and definitions .....	8
43	4 Symbols and abbreviated terms .....	10
44	5 Concepts .....	10
45	5.1 Interaction model for generic operations .....	10
46	5.2 Generic operations mappings .....	12
47	5.2.1 Overview .....	12
48	5.2.2 Recommendations .....	12
49	5.3 Conformance to generic operations .....	13
50	5.3.1 Conformance of entire WBEM protocols or APIs .....	13
51	5.3.2 Conformance of single WBEM operations or API calls .....	13
52	5.3.3 Requirement levels for operation parameters .....	14
53	5.4 Generic types .....	14
54	5.4.1 CIM data types .....	15
55	5.4.2 NamespacePath .....	15
56	5.4.3 InstancePath .....	15
57	5.4.4 ClassPath .....	15
58	5.4.5 QualifierTypePath .....	15
59	5.4.6 InstanceSpecification .....	15
60	5.4.7 ClassSpecification .....	16
61	5.4.8 QualifierType .....	17
62	5.4.9 InstanceSpecificationWithPath .....	17
63	5.4.10 ClassSpecificationWithPath .....	17
64	5.4.11 QualifierTypeWithPath .....	17
65	5.4.12 ClassName .....	18
66	5.4.13 PropertyName .....	18
67	5.4.14 MethodName .....	18
68	5.4.15 ParameterValue .....	18
69	5.4.16 ReturnValue .....	18
70	5.4.17 QueryString .....	18
71	5.4.18 QueryLanguage .....	18
72	5.4.19 EnumerationContext .....	18
73	5.4.20 ListenerDestination .....	18
74	5.5 Success and failure .....	19
75	5.6 Preconditions and postconditions .....	19
76	5.7 Generic error messages .....	19
77	5.8 Consistency model .....	20
78	5.8.1 Definition of ACID properties .....	20
79	5.8.2 Time consistency within instance representations .....	21
80	5.8.3 Staleness of information returned .....	21
81	5.8.4 Isolation between operations .....	21
82	5.8.5 Duplicate return of CIM objects or object paths .....	22
83	5.8.6 Time consistency between returned CIM objects .....	22
84	5.8.7 Order of returned CIM objects .....	22
85	5.8.8 Validity of returned object paths .....	23
86	5.8.9 Effects of deleting an instance .....	23

87	6	Generic operations .....	24
88	6.1	Description format.....	25
89	6.2	Common operation parameters for all operations .....	27
90	6.2.1	IncludeQualifiers .....	27
91	6.2.2	<element>List .....	27
92	6.3	Instance operations.....	27
93	6.3.1	GetInstance.....	27
94	6.3.2	DeleteInstance.....	29
95	6.3.3	ModifyInstance.....	32
96	6.3.4	CreateInstance.....	34
97	6.4	Instance enumeration operations .....	37
98	6.4.1	General behavioral rules.....	37
99	6.4.2	Common operation parameters for the open operations.....	39
100	6.4.3	OpenEnumerateInstances .....	41
101	6.4.4	OpenAssociators.....	45
102	6.4.5	OpenReferences.....	50
103	6.4.6	OpenQueryInstances .....	54
104	6.4.7	Common operation parameters for the pull operations .....	57
105	6.4.8	PullInstancesWithPath .....	58
106	6.4.9	PullInstances.....	60
107	6.4.10	CloseEnumeration .....	62
108	6.5	Method invocation operations .....	64
109	6.5.1	InvokeMethod .....	64
110	6.5.2	InvokeStaticMethod .....	66
111	6.6	Class operations .....	68
112	6.6.1	GetClass .....	68
113	6.6.2	DeleteClass.....	69
114	6.6.3	ModifyClass .....	73
115	6.6.4	CreateClass .....	75
116	6.7	Class enumeration operations .....	77
117	6.7.1	EnumerateClasses.....	77
118	6.7.2	AssociatorClasses .....	79
119	6.7.3	ReferenceClasses.....	82
120	6.8	Qualifier type operations .....	84
121	6.8.1	GetQualifierType.....	84
122	6.8.2	DeleteQualifierType .....	86
123	6.8.3	ModifyQualifierType.....	87
124	6.8.4	CreateQualifierType.....	89
125	6.8.5	EnumerateQualifierTypes .....	91
126	6.9	Indication delivery operations .....	92
127	6.9.1	DeliverIndication .....	92
128	ANNEX A (normative)	Cross-namespace associations.....	95
129	A.1	Binary association using same schema version.....	95
130	ANNEX B (informative)	Change log.....	99
131		Bibliography .....	102
132			

## 133 Figures

134	Figure 1 – Interaction model for generic server operations .....	11
135	Figure 2 – Interaction model for generic listener operations.....	11
136	Figure 3 – Generic operations mappings.....	12
137	Figure 4 – Typical profile representation of binary association crossing namespaces .....	95

138 Figure 5 – Binary association: WBEM server objects for bidirectional traversal ..... 96  
139 Figure 6 – Binary association: WBEM server objects for unidirectional traversal ..... 98

140

141 **Tables**

142 Table 1 – List of generic operations ..... 24

143

144

## Foreword

145 The *Generic Operations* specification (DSP0223) was originally prepared by the Generic Operations  
146 Working Group of the DMTF and is now owned by the Architecture Working Group of the DMTF.

147 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems  
148 management and interoperability. For information about the DMTF, see <http://www.dmtf.org>.

## 149 Acknowledgements

150 DMTF acknowledges the following individuals for their contributions to this specification:

- 151 • Jim Davis, WBEM Solutions
- 152 • George Ericson, EMC
- 153 • Steve Hand, Symantec
- 154 • Jon Hass, Dell
- 155 • Lawrence Lamers, VMware
- 156 • Andreas Maier, IBM (editor)
- 157 • Karl Schopmeyer, Inova Development

## 158 Document conventions

### 159 Typographical conventions

160 The following typographical conventions are used in this document:

- 161 • The titles of referenced documents are marked in *italics*.
- 162 • Important terms that are used for the first time are marked in *italics*.
- 163 • Generic parameters and generic types are marked in *italics*.
- 164 • The usage of terms typically links to their definition. Example: class path
- 165 • XML text is in `monospaced font`.

### 166 Experimental material

167 Experimental material has yet to receive sufficient review to satisfy the adoption requirements set forth by  
168 the DMTF. Experimental material is included in this document as an aid to implementers who are  
169 interested in likely future developments. Experimental material may change as implementation  
170 experience is gained. It is likely that experimental material will be included in an upcoming revision of the  
171 document. Until that time, experimental material is purely informational.

172 The following typographical convention indicates experimental material:

---

#### 173 **EXPERIMENTAL**

174 Experimental material appears here.

#### 175 **EXPERIMENTAL**

---

176 In places where this typographical convention cannot be used (for example, tables or figures), the  
177 "EXPERIMENTAL" label is used alone.

178

# Generic Operations

## 179 1 Scope

180 DMTF defines a number of protocols that describe how managed resources that are modeled using CIM  
181 can be discovered, accessed and manipulated:

- 182 • CIM-XML: The protocol defined in the *CIM Operations over HTTP Specification* ([DSP0200](#)), the  
183 *Representation of CIM in XML Specification* ([DSP0201](#)) and the *DTD for Representation of CIM*  
184 *in XML* ([DSP0203](#)).
- 185 • WS-Management: The usage of the WS-Management protocol for CIM, as defined in the *WS-*  
186 *Management CIM Binding Specification* ([DSP0227](#)), the *WS-CIM Mapping Specification*  
187 ([DSP0230](#)), the *Web Services for Management Specification* ([DSP0226](#)), and other underlying  
188 Web Services specifications.
- 189 • CIM-RS: The RESTful protocol for CIM, as defined in *CIM-RS Protocol* ([DSP0210](#)) and in *CIM-*  
190 *RS Payload Representation in JSON* ([DSP0211](#)).
- 191 • SM-CLP: The protocol defined in the *Server Management Command Line Protocol Specification*  
192 ([DSP0214](#)), covering the core of the protocol common for all management profiles, and SM-  
193 CLP mapping specifications for each management profile, covering profile-specific aspects of  
194 the protocol such as verbs for extrinsic methods.

195 As different as these protocols are, they have certain operations and semantics in common, at least when  
196 looking at it from a higher level. These common semantics can be used to define generic operations. This  
197 specification defines an operational model and behavior associated to these operations at an abstracted,  
198 generic level, and common across these protocols.

199 The generic operations are expected to be used in the following areas:

- 200 • Future releases of management profiles can define requirements on intrinsic operations by  
201 referencing generic operations. Currently, they do that by referencing the operations defined for  
202 the CIM-XML protocol. Using generic operations allows management profiles to become  
203 independent of protocols. Management profiles defined in XML using the *Management Profile*  
204 *XML Schema* ([DSP8028](#)) are required to use generic operations.
- 205 • Future and existing DMTF protocols can define mappings between their protocol-specific  
206 operations and the generic operations. This drives more commonality across these protocols,  
207 and consequently makes it easier to support multiple protocols in client applications, server side  
208 instrumentation, and mapping bridges between protocols (also known as protocol gateways).
- 209 • Client APIs, server APIs and provider APIs can define their API calls conformant to the generic  
210 operations. This drives more commonality across these APIs and between these APIs and  
211 WBEM protocols, and consequently makes it easier to support multiple protocols with the same  
212 API in client libraries and server side instrumentation (e.g., provider APIs).

## 213 2 Normative references

214 The following referenced documents are indispensable for the application of this specification. For dated  
215 or versioned references, only the edition cited (including any corrigenda or DMTF update versions)  
216 applies. For references without a date or version, the latest published edition of the referenced document  
217 (including any corrigenda or DMTF update versions) applies.

218 DMTF DSP0004, *CIM Infrastructure Specification 2.8*,  
219 [http://www.dmtf.org/standards/published\\_documents/DSP0004\\_2.8.pdf](http://www.dmtf.org/standards/published_documents/DSP0004_2.8.pdf)

220 DMTF DSP0198, *WBEM Glossary 1.0*,  
221 [http://www.dmtf.org/standards/published\\_documents/DSP0198\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0198_1.0.pdf)

222 DMTF DSP0207, *WBEM URI Mapping 1.0*,  
223 [http://www.dmtf.org/standards/published\\_documents/DSP0207\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0207_1.0.pdf)

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

226 DMTF DSP1054, *Indications Profile 1.2*,  
227 [http://www.dmtf.org/standards/published\\_documents/DSP1054\\_1.2.pdf](http://www.dmtf.org/standards/published_documents/DSP1054_1.2.pdf)

228 DMTF DSP8016, *WBEM Operations Message Registry 1.1*,  
229 [http://schemas.dmtf.org/wbem/messageregistry/1/dsp8016\\_1.1.xml](http://schemas.dmtf.org/wbem/messageregistry/1/dsp8016_1.1.xml)

230 ISO/IEC Directives, Part 2:2004, *Rules for the structure and drafting of International Standards*,  
231 <http://isotc.iso.org/livelink/livelink?func=ll&objId=4230456&objAction=browse>

## 232 **3 Terms and definitions**

233 In this specification, some terms have a specific meaning beyond the normal English meaning. Those  
234 terms are defined in this clause.

235 The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"),  
236 "may", "need not" ("not required"), "can" and "cannot" in this specification are to be interpreted as  
237 described in [ISO/IEC Directives, Part 2](#), Annex H. The terms in parenthesis are alternatives for the  
238 preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic  
239 reasons. [ISO/IEC Directives, Part 2](#), Annex H specifies additional alternatives. Occurrences of such  
240 additional alternatives shall be interpreted in their normal English meaning.

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

243 The terms "normative" and "informative" in this specification are to be interpreted as described in [ISO/IEC](#)  
244 [Directives, Part 2](#), Clause 3. In this specification, clauses, subclauses or annexes indicated with  
245 "(informative)" as well as notes and examples do not contain normative content.

246 The terms "class path", "creation class", "instance path", "management profile", "namespace path",  
247 "object", "object path", "qualifier type path", "WBEM client", "client", "WBEM listener", "listener", "WBEM  
248 server", "server", "WBEM operation", "WBEM protocol", and any other terms defined in [DSP0198](#) apply to  
249 this specification. The following additional terms are used in this document.

### 250 **3.1**

#### 251 **duplicate object**

252 objects in a result set that have duplicate object paths.

### 253 **3.2**

#### 254 **duplicate object path**

255 object paths in a result set that reference the same object accessible through the WBEM server.

### 256 **3.3**

#### 257 **effective qualifier value**

258 The effective value of a qualifier specified on a schema element is the value that determines the qualifier  
259 behavior for the schema element, taking the qualifier propagation rules into account. For a complete  
260 definition, see [DSP0004](#).



- 261 **3.4**  
262 **exposed elements of a class**  
263 The set of schema elements exposed by a class (i.e., properties and methods) is the union of the set of  
264 elements defined in the class (including overridden elements) and the set of inherited elements that are  
265 not overridden in the class. For a complete definition, see [DSP0004](#).
- 266 **3.5**  
267 **generic listener operation**  
268 a generic operation directed from a WBEM server to a WBEM listener. Also called listener operation. For  
269 details, see 5.1.
- 270 **3.6**  
271 **generic operation**  
272 a generic operation as defined in this specification. Also called operation. They are divided into generic  
273 listener operations and generic server operations. For details, see 5.1.
- 274 **3.7**  
275 **generic operation request**  
276 the request portion of a generic operation. Also called operation request. For details, see 5.1.
- 277 **3.8**  
278 **generic operation response**  
279 the response portion of a generic operation. Also called operation response. For details, see 5.1.
- 280 **3.9**  
281 **generic operations mapping**  
282 a mapping of generic operations to the operations of some other protocol (e.g., WBEM operations) or to  
283 the calls of some API, as defined in 5.2.
- 284 **3.10**  
285 **generic server operation**  
286 a generic operation directed from a WBEM client to a WBEM server. Also called server operation. For  
287 details, see 5.1.
- 288 **3.11**  
289 **isolation**  
290 the set of behaviors that describe how the execution of an operation affects the execution of another,  
291 concurrent operation, as defined in 5.8.4.
- 292 **3.12**  
293 **volatile property**  
294 a property in an instance whose value may change as a WBEM client obtains the instance repeatedly  
295 without performing any client-originated updates to the property value.
- 296 **3.13**  
297 **WBEM listener operation**  
298 a WBEM operation that is originated on a WBEM server and processed by a WBEM listener. For details,  
299 see 5.1.
- 300 **3.14**  
301 **WBEM protocol mapping**  
302 a mapping of generic operations to a WBEM protocol, as defined in 5.2.

303 **3.15**

304 **WBEM server operation**

305 a WBEM operation that is originated by a WBEM client and processed by a WBEM server. For details,  
306 see 5.1.

307 **4 Symbols and abbreviated terms**

308 The abbreviations "API", "CIM", "CIM-XML", "CIM-RS", "CQL", "UML", "WBEM", "WS-Management",  
309 "XML", and any other symbols and abbreviations defined in [DSP0198](#) apply to this specification. The  
310 following additional abbreviations are used in this document.

311 **4.1**

312 **SM-CLP**

313 Server Management Command Line Protocol, defined in [DSP0214](#)

314 **5 Concepts**

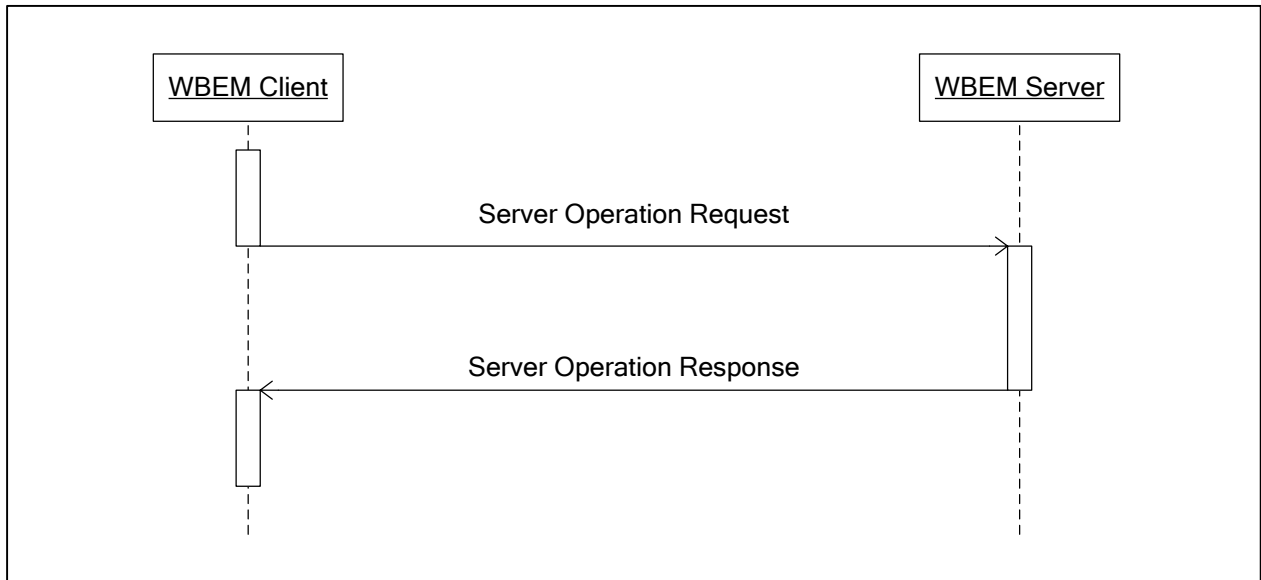
315 This clause defines concepts that are the basis for the definition of the generic operations.

316 **5.1 Interaction model for generic operations**

317 Generic operations are divided into two categories:

- 318 • **Generic server operations:** An operation request is sent from a WBEM client to a WBEM  
319 server in order to initiate the processing of the operation, and an operation response is sent  
320 back from the server to the client upon completion of the operation.
- 321 • **Generic listener operations:** An operation request is sent from a WBEM server to a WBEM  
322 listener in order to initiate the processing of the operation, and an operation response is sent  
323 back from the listener to the server upon completion of the operation.

324 Figure 1 shows the interaction model for generic server operations, using a UML sequence diagram:

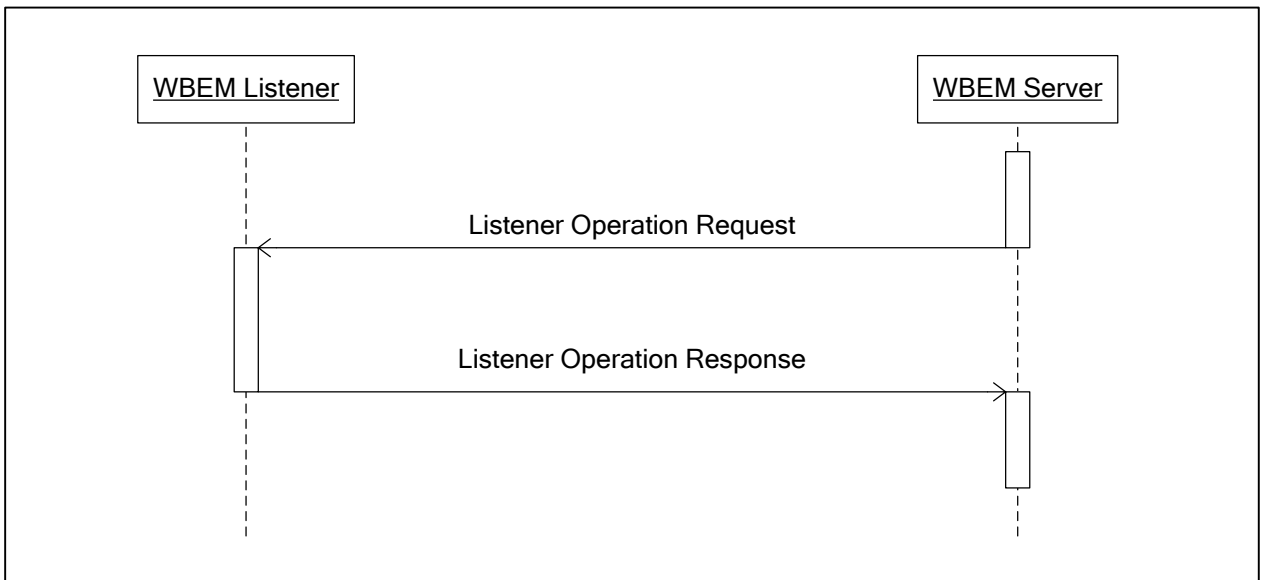


325  
326

**Figure 1 – Interaction model for generic server operations**

327

328 Figure 2 shows the interaction model for generic listener operations, using a UML sequence diagram:



329  
330

**Figure 2 – Interaction model for generic listener operations**

331

332 The operation request and operation response at the level of generic operations do not necessarily need  
 333 to correspond directly to WBEM operations, that is to messages that are flowing at the level of the WBEM  
 334 protocol. For example, a generic operation response may be delivered asynchronously at the level of the  
 335 WBEM protocol.

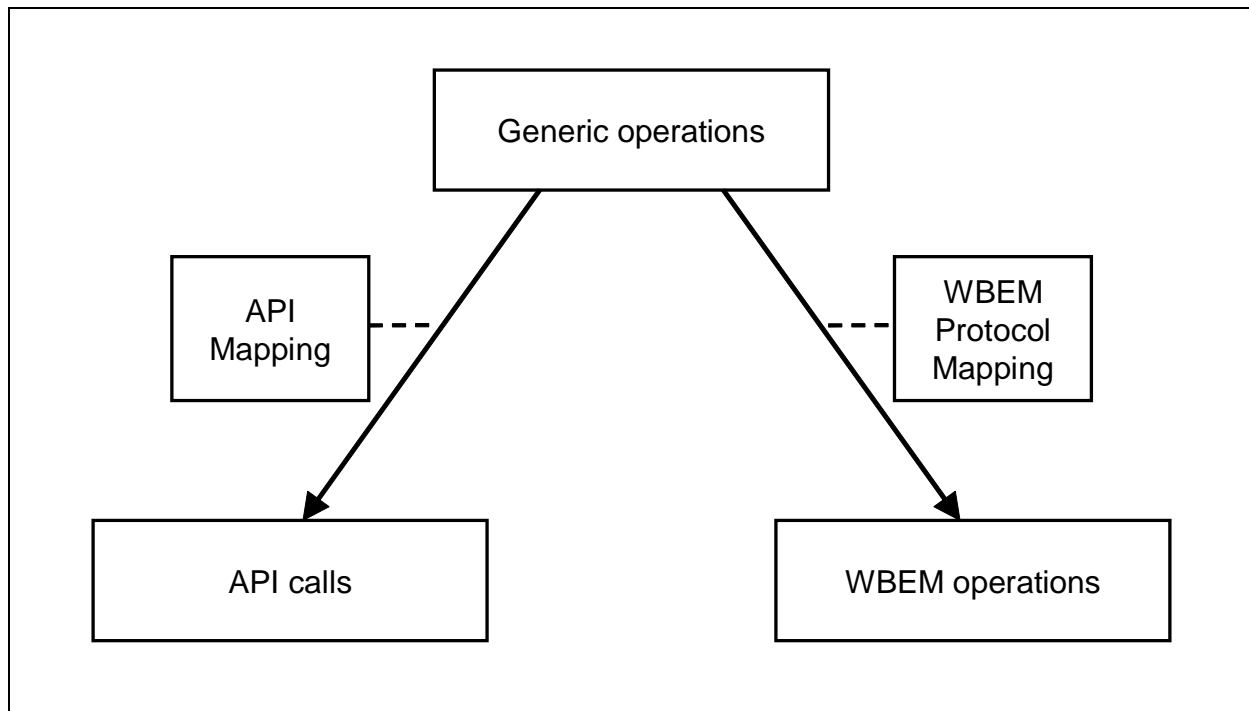
336 At the level of generic operations, any *input parameters* are part of the operation request, and any *output*  
 337 *parameters* are part of the operation response. A WBEM protocol may choose to do that differently, for  
 338 example by pushing some of the input parameters to the server in the form of options that are set, and  
 339 that are used during the processing of subsequent operations.

340 This abstraction of generic operations from WBEM operations allows keeping the definition of the generic  
 341 operations simple and scoped to defining the operation semantics. The details about the actual message  
 342 flows are left to the scope of WBEM protocols. This separation is key in order to use the same definition  
 343 of generic operations for multiple WBEM protocols.

344 **5.2 Generic operations mappings**

345 **5.2.1 Overview**

346 Figure 3 shows mappings of generic operations to WBEM protocols and APIs. These mappings allow  
 347 determining which WBEM operations or API calls need to be implemented for a particular generic  
 348 operation to be supported. This is used for example when implementing management profiles that specify  
 349 provisions for intrinsic operations by referencing generic operations.



350

351 **Figure 3 – Generic operations mappings**

352 **5.2.2 Recommendations**

353 This subclause provides recommendations for specifying WBEM protocol mappings and API mappings  
 354 that provide for determining the WBEM operations or API calls that support a particular generic operation,  
 355 and specify conformance.

356 There is no requirement that WBEM protocol mappings and API mappings are defined in a separate  
 357 specification (i.e., they can be defined in the specifications that define the WBEM protocol or API).

358 The following recommendations apply:

- 359 • WBEM protocol mappings and API mappings should define the mapping from a perspective of  
360 the generic operation (i.e., by listing the relevant generic operation at the top level).
- 361 • For each generic operation listed in the mapping, the corresponding WBEM operations or API  
362 calls should be stated that provide the functionality supporting the generic operation.
- 363 • For each parameter defined for a generic operation listed in the mapping, the corresponding  
364 parameters and return values of the WBEM operations or API calls should be stated.
- 365 • A statement should be made for each generic operation as to whether or not the operation is  
366 supported in a conformant way, as defined in 5.3.2. If the operation is supported in a non-  
367 conformant way, the deviations should be stated.
- 368 • A statement should be made for the entire WBEM protocol or API as to whether or not it is  
369 conformant to generic operations.

### 370 5.3 Conformance to generic operations

371 Conformance of a WBEM protocol or API to generic operations is defined at two levels:

- 372 1) At the level of the entire WBEM protocol or API
- 373 2) At the level of single WBEM operations or single API calls

374 The guiding principle for conformance to generic operations is that a WBEM protocol or API call is able to  
375 completely represent the generic operations and their semantics. Functionalities of the WBEM protocol or  
376 API that go beyond the functionality of generic operations are not relevant for conformance.

#### 377 5.3.1 Conformance of entire WBEM protocols or APIs

378 A WBEM protocol or API is conformant to generic operations if all generic operations defined in this  
379 specification are supported by WBEM operations or API calls in a conformant way, as defined in 5.3.2.

380 Conformant WBEM protocols or APIs may define WBEM operations or API calls in addition to those that  
381 are mapped to generic operations.

#### 382 5.3.2 Conformance of single WBEM operations or API calls

383 A particular generic operation is supported by WBEM operations or API calls in a conformant way if all of  
384 the following is satisfied:

- 385 • The generic operation has one or more corresponding WBEM operations or API calls that  
386 provide the functionality of the generic operation. The names of these corresponding WBEM  
387 operations or API calls may be different from the name of the generic operation.
- 388 • Functionalities that are required to be supported for a generic operation are supported by the  
389 corresponding WBEM operations or API calls with the semantics defined by the generic  
390 operation.
- 391 • If functionalities that are optional to be supported for a generic operation are supported by the  
392 corresponding WBEM operations or API calls, they are supported with the semantics defined by  
393 the generic operation.
- 394 • Each parameter of a generic operation is mapped to one or more corresponding parameters of  
395 the corresponding WBEM operations or API calls
- 396 • For each parameter of a generic operation, the provisions defined in 5.3.3 are satisfied.

397 WBEM operations or API calls that support a generic operation in a conformant way, may support  
398 parameters or return values in addition to the parameters mapped to parameters of the corresponding

399 generic operation. Defining additional parameters can affect the ability to transform one WBEM protocol  
400 into another (e.g. in protocol gateways).

### 401 **5.3.3 Requirement levels for operation parameters**

402 The parameters defined for generic operations each have a requirement level, as defined in this  
403 subclause.

404 A conformant WBEM protocol or API shall delegate these requirement levels to the receiver of the WBEM  
405 operation (for example, to the WBEM server for a server operation, and to the WBEM listener for a  
406 listener operation), as follows:

#### 407 **Mandatory**

408 For operation parameters designated as mandatory, conformant WBEM protocols and APIs  
409 shall document the support of the corresponding operation parameters by the receiver of the  
410 WBEM operation as:

- 411 • Mandatory.

#### 412 **Conditional**

413 For operation parameters designated as conditional, conformant WBEM protocols and APIs  
414 shall document the support of the corresponding operation parameters by the receiver of the  
415 WBEM operation as one of:

- 416 • Mandatory, if the condition can be evaluated at the time the WBEM protocol or API  
417 specification is written and the condition is met,
- 418 • Optional, if the condition can be evaluated at the time the WBEM protocol or API  
419 specification is written and the condition is not met,
- 420 • Conditional, potentially with a different condition, if the condition cannot be evaluated at the  
421 time the WBEM protocol or API specification is written.

#### 422 **Optional**

423 For operation parameters designated as optional, conformant WBEM protocols and APIs shall  
424 document the support of the corresponding operation parameters by the receiver of the WBEM  
425 operation as one of:

- 426 • Mandatory,
- 427 • Optional,
- 428 • Conditional.

429 In all those cases, conformant WBEM protocols and APIs may specify that supplying values for a  
430 supported parameter is optional as long as the protocol or API defines a default value for the parameter.  
431 In other words, there are two different kinds of requirements related to parameters:

- 432 1) The requirement defined by this document to support a parameter in a WBEM protocol or API  
433 by the receiver of an operation
- 434 2) The requirement defined by the WBEM protocol or API specification for supplying a value for a  
435 supported parameter by the originator of an operation

### 436 **5.4 Generic types**

437 This specification defines the following generic data types for use by operation parameters of generic  
438 operations.

#### 439 5.4.1 CIM data types

440 All CIM data types defined in [DSP0004](#) (e.g., boolean) may be used as generic types. Values of these  
441 data types can assume the (untyped) value NULL, as defined in [DSP0004](#).

#### 442 5.4.2 NamespacePath

443 A value of the generic type *NamespacePath* represents a namespace path as defined in [DSP0004](#).

444 This specification does not define particular sub-components of a namespace path; as a result, any  
445 requirements on the presence of such sub-components are left to conformant WBEM protocols.

446 Conformant WBEM protocols shall support all characteristics of *NamespacePath* values and may support  
447 additional characteristics.

#### 448 5.4.3 InstancePath

449 A value of the generic type *InstancePath* represents an instance path as defined in [DSP0004](#).

450 An *InstancePath* value shall specify the class name and key binding components of the represented  
451 instance path. Any requirements for specifying or omitting the namespace path component in an  
452 *InstancePath* value are left to conformant WBEM protocols.

453 Conformant WBEM protocols shall support all characteristics of *InstancePath* values and may support  
454 additional characteristics.

#### 455 5.4.4 ClassPath

456 A value of the generic type *ClassPath* represents a class path as defined in [DSP0004](#).

457 A *ClassPath* value shall specify the class name component of the represented class path. Any  
458 requirements for specifying or omitting the namespace path component in a *ClassPath* value are left to  
459 conformant WBEM protocols.

460 Conformant WBEM protocols shall support all characteristics of *ClassPath* values and may support  
461 additional characteristics.

#### 462 5.4.5 QualifierTypePath

463 A value of the generic type *QualifierTypePath* represents a qualifier type path as defined in [DSP0004](#).

464 A *QualifierTypePath* value shall specify the qualifier name component of the represented qualifier type  
465 path. Any requirements for specifying or omitting the namespace path component in a *QualifierTypePath*  
466 value are left to conformant WBEM protocols.

467 Conformant WBEM protocols shall support all characteristics of *ClassPath* values may support additional  
468 characteristics.

#### 469 5.4.6 InstanceSpecification

470 A value of the generic type *InstanceSpecification* is a representation of a CIM instance as defined for the  
471 *Instance* meta-element defined in [DSP0004](#), containing:

- 472
- 473 • name of the creation class of the instance
  - 474 • all or a subset of the static and non-static properties exposed by the creation class of the instance

475 Each property in an *InstanceSpecification* shall contain:

- 476 • name of the property
- 477 • value of the property
- 478 • optional: Class origin of the property
- 479 • optional: Data type of the property

480 *InstanceSpecification* does not contain the instance path of the instance, because there are some  
481 situations in which the instance data is needed without an instance path. The  
482 *InstanceSpecificationWithPath* type is used when the instance path is needed in addition to the instance  
483 data.

484 Generic operations using this type define the rules for the optional items in the content of this type.

#### 485 **5.4.7 ClassSpecification**

486 A value of the generic type *ClassSpecification* is a representation of a CIM class as defined for the *Class*  
487 meta-element defined in [DSP0004](#), containing:

- 488 • name of the class
- 489 • name of the superclass, if any
- 490 • all or a subset of the static and non-static properties (that is, the property definitions) exposed  
491 by the class. As defined in DSP0004, the set of properties exposed by a class includes any  
492 properties inherited from superclasses, where overridden properties are included only once.
- 493 • all of the static and non-static methods exposed by the class. As defined in DSP0004, the set of  
494 methods exposed by a class includes any methods inherited from superclasses, where  
495 overridden methods are included only once.
- 496 • optional: all of the qualifiers exposed by the class that are defined on the class or any of its  
497 superclasses

498 Each property in a *ClassSpecification* shall contain:

- 499 • name of the property
- 500 • data type of the property
- 501 • default value of the property
- 502 • optional: all of the qualifiers exposed by the property that are defined on the property or any of  
503 its overridden properties

504 Each method in a *ClassSpecification* shall contain:

- 505 • name of the method
- 506 • data type of the return value of the method
- 507 • all of the parameters of the method
- 508 • optional: all of the qualifiers exposed by the method that are defined on the method or any of its  
509 overridden methods



510 Each parameter in that method shall contain:

- 511 • name of the parameter
- 512 • data type of the parameter
- 513 • optional: all of the qualifiers exposed by the parameter that are defined on the parameter or the
- 514 corresponding parameter in any of its overridden methods

515 Each qualifier in any of the items above shall contain:

- 516 • name of the qualifier
- 517 • effective value of the qualifier, as seen in the scope of the class represented by *Class*

518 *ClassSpecification* does not contain the class path of the class. The *ClassSpecificationWithPath* type is  
519 used when the class path is needed in addition to the class.

520 Generic operations using this type define the rules for the optional items in the content of this type.

#### 521 **5.4.8 QualifierType**

522 A value of the generic type *QualifierType* is a representation of a CIM qualifier type as defined for the  
523 *QualifierType* meta-element defined in [DSP0004](#) (i.e., a qualifier declaration) containing:

- 524 • name of the qualifier
- 525 • data type of the qualifier
- 526 • default value of the qualifier
- 527 • all flavors of the qualifier
- 528 • all scopes of the qualifier

529 *QualifierType* does not contain the qualifier type path of the qualifier type. The *QualifierTypeWithPath*  
530 type is used when the qualifier type path is needed in addition to the qualifier type.

#### 531 **5.4.9 InstanceSpecificationWithPath**

532 A value of the generic type *InstanceSpecificationWithPath* combines the content of *InstanceSpecification*  
533 and *InstancePath*.

534 *InstanceSpecification* shall represent the instance referenced by *InstancePath*.

#### 535 **5.4.10 ClassSpecificationWithPath**

536 A value of the generic type *ClassSpecificationWithPath* combines the content of *ClassSpecification* and  
537 *ClassPath*.

538 *ClassSpecification* shall represent the class referenced by *ClassPath*.

#### 539 **5.4.11 QualifierTypeWithPath**

540 A value of the generic type *QualifierTypeWithPath* combines the content of *QualifierType* and  
541 *QualifierTypePath*.

542 *QualifierType* shall represent the qualifier type referenced by *QualifierTypePath*.

**543 5.4.12 ClassName**

544 A value of the generic type *ClassName* is the name of a CIM class, including its schema prefix.

**545 5.4.13 PropertyName**

546 A value of the generic type *PropertyName* is the name of a CIM property or reference.

547 The class defining the property is not identified by the data in this type.

**548 5.4.14 MethodName**

549 A value of the generic type *MethodName* is the name of a CIM method.

550 The class defining the method and the method signature are not identified by the data in this type.

**551 5.4.15 ParameterValue**

552 A value of the generic type *ParameterValue* is a parameter value used as an input or output parameter  
553 during invocation of a CIM method, containing:

- 554 • name of the parameter
- 555 • value of the parameter
- 556 • optional: Data type of the parameter

557 Generic operations using this type define the rules for the optional items in the content of this type.

**558 5.4.16 ReturnValue**

559 A value of the generic type *ReturnValue* is the value returned by the invocation of a CIM method,  
560 containing:

- 561 • return value
- 562 • optional: Data type of the return value

563 Generic operations using this type define the rules for the optional items in the content of this type.

**564 5.4.17 QueryString**

565 A value of the generic type *QueryString* is a query string in some query language. The query language is  
566 not identified by the data in this type.

**567 5.4.18 QueryLanguage**

568 A value of the generic type *QueryLanguage* is a query language of a query string.

**569 5.4.19 EnumerationContext**

570 A value of the generic type *EnumerationContext* is a value that uniquely identifies an enumeration  
571 session used in pulled instance enumeration operations. It is opaque to WBEM clients.

**572 5.4.20 ListenerDestination**

573 A value of the generic type *ListenerDestination* is a value that uniquely addresses a WBEM listener for  
574 purposes of delivering an indication to it using the *DeliverIndication* operation (see 6.9.1).

575 The format of the address is defined by the WBEM protocol.

## 576 **5.5 Success and failure**

577 All generic operations either succeed or fail. There is no concept of "partial success".

578 If a generic operation succeeds, it delivers its output data back to the operation requester, and does not  
579 include any error messages.

580 If it fails, it delivers back one or more error messages, and no output data. For details about error  
581 messages, see 5.7.

582 For example, if an instance enumeration operation were able to return some instances successfully, but  
583 not all successfully, then the operation shall fail without returning any instances.

584 The WBEM operations mapped to generic operations by a conformant WBEM protocol shall also either  
585 succeed or fail, as described above.

## 586 **5.6 Preconditions and postconditions**

587 Each generic operation specifies a set of zero or more preconditions and a set of zero or more  
588 postconditions.

589 Each precondition in the set needs to be satisfied for the operation to be able to succeed. If one or more  
590 preconditions are not satisfied, the operation shall fail, indicating the unsatisfied precondition using a  
591 generic error message from the set listed for the operation that describes the unsatisfied precondition.

592 A successful execution of the generic operation shall guarantee that all postconditions in the set are  
593 satisfied.

## 594 **5.7 Generic error messages**

595 Each generic operation specifies a set of generic error messages. These generic error messages are  
596 DMTF standard messages (see [DSP0228](#)) from the WBEM Operations Message Registry ([DSP8016](#)).  
597 Each error message from this registry describes a particular error situation.

598 A conformant WBEM protocol shall support error handling in the following way:

- 599 • Return DMTF standard messages (also known as "extended error handling").  
600 In this case, the WBEM operation shall return the generic error message defined for the generic  
601 operation that matches the error situation, and may return additional error messages.

602 The other alternatives permitted in version 1 of this document are no longer permitted.

603 The generic error messages specified for each generic operation have a requirement level defined in  
604 context of that operation.

605 A conformant WBEM protocol or API shall delegate these requirement levels to the receiver of the WBEM  
606 operation (for example, to the WBEM server for a server operation, and to the WBEM listener for a  
607 listener operation), as follows.

608 The allowable requirement levels for generic error messages in the context of a generic operation are:

### 609 **Mandatory**

610 For generic error messages designated as mandatory, conformant WBEM protocols and APIs  
611 shall document the error message as mandatory to be supported by the receiver of the WBEM  
612 operation.

**613 Conditional**

614 For generic error messages designated as conditional, conformant WBEM protocols and APIs  
615 shall document the support of the error message by the receiver of the WBEM operation as one  
616 of:

- 617 • Mandatory, if the condition can be evaluated at the time the WBEM protocol or API  
618 specification is written and the condition is met,
- 619 • Optional, if the condition can be evaluated at the time the WBEM protocol or API  
620 specification is written and the condition is not met,
- 621 • Conditional, potentially with a different condition, if the condition cannot be evaluated at the  
622 time the WBEM protocol or API specification is written.

**623 Optional**

624 For generic error messages designated as optional, conformant WBEM protocols and APIs  
625 shall document the support of the error message by the receiver of the WBEM operation as one  
626 of:

- 627 • Mandatory,
- 628 • Optional,
- 629 • Conditional.

630 Each generic operation designates one of its input parameters to be a "context parameter." The  
631 messages defined in the WBEM Operations Message Registry ([DSP8016](#)) may include name and value  
632 of the context parameter in order to provide information about the invocation context.

633 This specification does not define any order or precedence for generic error messages to be returned by  
634 generic operations. This implies that the order in which the generic error messages are listed in the  
635 description of each generic operation has no binding significance on the order in which a conformant  
636 WBEM protocol would need to apply any tests to surface these errors, nor does the documented order  
637 require a precedence of error messages. However, the order in which the generic error messages are  
638 listed is meant to give some guidance about a typical order of precedence.

639 WBEM clients shall be prepared to deal with all generic error messages that are listed for a generic  
640 operation.

**641 5.8 Consistency model**

642 This subclause defines consistency requirements for generic operations.

643 Conformant WBEM protocols shall conform to the rules defined in this subclause for the WBEM  
644 operations to which the supported generic operations are mapped. WBEM protocols may define  
645 additional constraints for WBEM operations.

646 This specification does not define responsibilities for detecting violations to these rules.

**647 5.8.1 Definition of ACID properties**

648 This subclause defines atomicity, consistency, isolation and durability (ACID) properties for use by  
649 generic operations defined in this specification and by management profiles (see [DSP1001](#)).

650 Each generic operation defines requirements on its ACID properties. Management profiles that use  
651 generic operations to state their operation requirements inherit these requirements on ACID properties  
652 and may specify additional requirements. Profiles should not remove or weaken requirements on ACID  
653 properties defined by generic operations.

**654 5.8.1.1 Atomicity**

655 Operations and methods are considered *atomic* if and only if their effects on the managed resources and  
656 on CIM instances either occur completely or not at all.

657 Atomicity only applies to operations and methods that modify the managed resources or CIM instances  
658 through the management interface.

**659 5.8.1.2 Update consistency**

660 Operations and methods are considered *update-consistent* if and only if the managed resources and CIM  
661 instances are never left in an inconsistent state after a modification.

662 What constitutes a consistent state is defined in [DSP0004](#) and in management profiles.

663 Update consistency only applies to operations and methods that modify the managed resources or CIM  
664 instances through the management interface.

**665 5.8.1.3 Isolation**

666 Operations and methods are considered *isolated* if and only if their results and their effects on the  
667 managed resources and on CIM instances appear to be serialized with the results and effects of any  
668 other operations and methods, as observed through the management interface.

669 Isolation applies to operations and methods that retrieve information through the management interface,  
670 and to operations that modify the managed resources or CIM instances through the management  
671 interface.

**672 5.8.1.4 Durability**

673 Operations and methods are considered *durable* if and only if their effects on the managed resources and  
674 on CIM instances will not be undone, other than by some other action that may or may not be caused  
675 through the profile defined management interface.

676 Durability only applies to operations and methods that modify the managed resources or CIM instances  
677 through the management interface.

**678 5.8.2 Time consistency within instance representations**

679 The property values of an instance representation returned by any generic operation shall represent a  
680 snapshot of the instance object that exists in the server.

681 If a WBEM protocol provides the capability to transfer an operation response in multiple parts, and a  
682 particular instance representation is distributed over multiple parts of the response which are transferred  
683 at different points in times, the property values of that instance representation still need to satisfy the time  
684 consistency constraint.

**685 5.8.3 Staleness of information returned**

686 Conformant WBEM protocols should define that implementations should do a best effort to return the  
687 most current information, as far as property values of instances and also the existence of instances are  
688 concerned.

**689 5.8.4 Isolation between operations**

690 This specification defines no particular requirements regarding isolation between operations in addition to  
691 the other consistency rules defined in 5.8.

692 For example, if an instance is deleted and after that another one is created, an enumeration operation  
693 executed concurrently may consistently include the instance that got deleted just before that happened,  
694 as well as the new instance after it got consistently created, hence returning a set of instances that never  
695 existed at the same time. This example satisfies all consistency rules defined in this specification.  
696 An example where other consistency rules determine the overall behavior is a GetInstance operation  
697 executing concurrently with a

698 ModifyInstance operation on the same instance. The consistency rules defined in 5.8.2 require that this  
699 GetInstance operation needs to return an instance representation that either has none or all of the  
700 modifications requested by the ModifyInstance operation.

### 701 **5.8.5 Duplicate return of CIM objects or object paths**

702 Any generic operations returning CIM object representations or CIM object paths should not return  
703 duplicate objects or duplicate object paths.

704 If duplicate objects or duplicate object paths are returned, WBEM clients should consider the last  
705 occurrence of a duplicate object or duplicate object path in the sequence as the valid occurrence to work  
706 with, and should ignore all other duplicate occurrences.

707 [DSP0004](#) requires that a CIM namespace in a WBEM server does not contain duplicate objects (i.e.,  
708 instances, classes, qualifier types) at any point in time. However, given the rule above, the result set of a  
709 generic operation may.

710 An example for a situation in which duplicate instances or instance paths might be returned is a sequence  
711 of instance deletion and creation with the same key values concurrently to an enumeration operation, all  
712 in the same namespace.

713 As a consequence, a WBEM server is not obliged to test for, correct or reject any duplicate objects or  
714 object paths in the result set of an operation.

### 715 **5.8.6 Time consistency between returned CIM objects**

716 This specification does not mandate any time consistency between the CIM objects or CIM object paths  
717 returned by generic operations.

718 For example, if a WBEM server processes an instance enumeration operation by contacting multiple  
719 independent infrastructure components each of which contributes instances to the combined result set,  
720 the result set may contain instance representations that represent different points in time.

721 However, the rule defined in 5.8.2 requires that consistency is maintained within each single instance  
722 representation.

### 723 **5.8.7 Order of returned CIM objects**

724 For operations that do not support the specification of a sort order, the order of returned CIM objects is  
725 implementation-dependent.

726 For example, if a WBEM server processes an instance enumeration operation by contacting multiple  
727 independent infrastructure components each of which contributes instances to the combined result set,  
728 the resulting order might be an arbitrary merge of the sequences of instances contributed by each  
729 component.

730 WBEM protocols may define additional requirements on the order of returned CIM objects.

731 **5.8.8 Validity of returned object paths**

732 This specification does not mandate that object paths returned to a WBEM client are still valid by the time  
733 the WBEM client attempts to use them in subsequent operations in order to address those objects.

734 For example: if a WBEM server returns an instance path and an operation then deletes the instance, a  
735 subsequent attempt to get the instance using the returned instance path will fail.

736 **5.8.9 Effects of deleting an instance**

737 Deleting an instance may affect the overall consistency because other instances depend on the instance  
738 to be deleted. Instances that depend on the instance to be deleted are called "dependent instances" in  
739 this specification.

740 The behavior of operations that delete instances (such as *DeleteInstance*) cannot be defined in a  
741 generally applicable way. The following options are available for defining the handling of the deletion of  
742 an instance in the presence of dependent instances (e.g., in management profiles or in the CIM schema):

- 743 • **Delete propagation:** Delete any dependent instances implicitly along with the instance to be  
744 deleted.

745 Specifications using this specification need to give particular consideration to circular  
746 dependencies when defining rules for propagating deletion.

747 NOTE Such dependent instances may reside in a different namespace (which may reside in a different  
748 WBEM server) than the instance to be deleted.

- 749 • **Rejection:** Reject the deletion of the instance to be deleted, leaving it to the WBEM client to  
750 delete dependent instances first.

751 The following options are **not** available for defining the handling of the deletion of an instance in the  
752 presence of dependent instances:

- 753 • **Deletion without propagation:** Delete the instance to be deleted but do not delete any  
754 dependent instances. This would cause an inconsistent state in the model.

755 The following instances are considered dependent instances for this purpose:

- 756 • **Composition:** Instances associated to an instance to be deleted, via a composition where the  
757 instance to be deleted is on the aggregate side.

758 The definition of the *Composition* qualifier in [DSP0004](#) requires that this case is handled by  
759 propagating the deletion of the aggregate instance to any aggregated instances and their  
760 composition instances.

- 761 • **Key propagation:** Instances of classes that have propagated keys (key properties exposing a  
762 value of TRUE for the *Propagated* qualifier, i.e., weak instances) are considered dependents of  
763 the instance from which the keys propagate (i.e., the strong instance).

764 The definition of the *Propagated* qualifier in [DSP0004](#) requires that this case is handled by  
765 propagating the deletion of the strong instance to any weak instances and their association  
766 instances.

- 767 • **Referencing associations:** Association instances that reference the instance to be deleted.

768 This case shall be handled with any or a combination of the following options:

- 769 – by propagating the deletion of the referenced instance to its referencing association  
770 instance
- 771 – by rejecting the deletion of the referenced instance to be deleted.

772 • **Qualifier defined delete propagation:** Instances to be deleted as a result of *IfDelete* and  
 773 *Delete* qualifiers, as defined in [DSP0004](#).

774 Support of the *IfDelete* and *Delete* qualifiers by a WBEM server is optional, as defined in  
 775 [DSP0004](#).

776 This concept can be used to propagate deletion from an instance to its referencing association  
 777 instance, from an association instance to its referenced instances, and in combination also  
 778 between associated instances.

779 The definition of the *IfDelete* and *Delete* qualifiers in [DSP0004](#) requires that this case is handled  
 780 by propagating the deletion of an instance to which the *IfDelete* qualifier applies, to any  
 781 instances to which the corresponding *Delete* qualifier applies.

782 • **Multiplicity underflow:** Instances associated to an instance to be deleted via an association  
 783 with a minimum multiplicity (as defined with *Min* qualifier in the schema, or as constrained by  
 784 management profiles) larger than 0 on the reference to the instance to be deleted, if the deletion  
 785 would violate the minimum multiplicity that is required.

786 EXAMPLE: Association AB references class A with *Min* (2) and references class B. Therefore, each  
 787 instance of B is supposed to be associated via AB with least two instances of A. If an instance of A is to  
 788 be deleted, and there is only one other instance of A associated to the instance of B that is associated  
 789 with the instance of A to be deleted, the minimum multiplicity would be violated by the deletion.

790 This case shall be handled with any or a combination of the following options:

- 791 – by propagating the deletion of the instance to be deleted to its associated instance defining  
 792 the multiplicity constraint, and the association instance.
- 793 – by rejecting the original deletion.

## 794 6 Generic operations

795 This clause defines the generic operations. They are listed in Table 1, grouped by their headings.

796 **Table 1 – List of generic operations**

Group	Generic Operation	Description
Instance operations	GetInstance	See 6.3.1
	DeleteInstance	See 6.3.2
	ModifyInstance	See 6.3.3
	CreateInstance	See 6.3.4
Instance enumeration operations	OpenEnumerateInstances	See 6.4.3
	OpenAssociators	See 6.4.4
	OpenReferences	See 6.4.5
	OpenQueryInstances	See 6.4.6
	PullInstancesWithPath	See 6.4.8
	PullInstances	See 6.4.9
Method invocation operations	CloseEnumeration	See 6.4.10
	InvokeMethod	See 6.5.1
	InvokeStaticMethod	See 6.5.2



Group	Generic Operation	Description
Class operations	GetClass	See 6.6.1
	DeleteClass	See 6.6.2
	ModifyClass	See 6.6.3
	CreateClass	See 6.6.4
Class enumeration operations	EnumerateClasses	See 6.7.1
	AssociatorClasses	See 6.7.2
	ReferenceClasses	See 6.7.3
Qualifier type operations	GetQualifierType	See 6.8.1
	DeleteQualifierType	See 6.8.2
	ModifyQualifierType	See 6.8.3
	CreateQualifierType	See 6.8.4
	EnumerateQualifierTypes	See 6.8.5

797

798 **6.1 Description format**

799 The generic operations are described using the following format. Items in angle brackets (e.g., "<name>")  
 800 need to be replaced by some other text, as described further down in this subclause.

801 **Purpose:**

802 <Short description of the purpose of the operation.>

803 **Operation Input Parameters:**

804

Generic Name	Generic Type	Requirement	Description
<diname>	<ditype>	<direq>	<Description of the operation parameter, including any conditions for requirement level Conditional> <The text "(Context Parameter)" for the parameter that is supposed to be displayed in messages, as defined in 5.7>
...	...	...	...

805

806 **Operation Output Parameters:**

807

Generic Name	Generic Type	Requirement	Description
<diname>	<ditype>	<direq>	<Description of the operation parameter, including any conditions for requirement level Conditional>
...	...	...	...

808

809 **Description:**

810 <A detailed description of the semantics of the operation including all conditions and behaviors  
 811 except those listed under Preconditions and Postconditions>

812 **Preconditions:**

813 • <List of additional preconditions for the operation, in plain text. Preconditions pertain to the state  
 814 before an operation gets invoked. They have nothing to do with the execution of the operation  
 815 or any effects the operation causes. They represent the conditions that are required to be met in  
 816 order for the operation to have a chance to execute successfully. Although not required for  
 817 preconditions, this specification uses "shall" to specify preconditions.>

818 **Postconditions:**

819 • <List of additional postconditions for the operation, in plain text. Postconditions describe the  
 820 state after an operation has been executed successfully. In other words, they represent the  
 821 guarantees an implementation needs to give in the case of successful execution.>

822 **Error messages:**

823

Message ID	Message Name	Requirement	Sources	Additional Description
<msgid>	<msgname>	<msgreq>	<msgsrc>	<Any description in addition to the description in the message registry>
...	...	...		...

824

825 The items in angle brackets that are not already described in the format above, have the following  
 826 meaning:

- 827 <diname> Generic name of the operation parameter.
- 828 <ditype> Generic type of the operation parameter, as defined in 5.4.
- 829 <direq> Requirement level of the operation parameter, as defined in 5.3.3.
- 830 <msgid> Message ID of the message, as defined in a DMTF message registry. The message  
 831 ID is the concatenation of the values of the XML attributes  
 832 MESSAGE/MESSAGE\_ID@PREFIX and  
 833 MESSAGE/MESSAGE\_ID@SEQUENCE\_NUMBER.
- 834 <msgname> Message name of the message, as defined in a DMTF message registry. The  
 835 message name is the value of the XML attribute MESSAGE@NAME.
- 836 <msgreq> Requirement level of the message, as defined in 5.7.

837 <msgsrc> Sources of the message. One or more values may be specified. Valid values are:  
838 Infrastructure – the message is implemented by the common infrastructure portion  
839 of the WBEM server.  
840 Class implem. – the message is implemented by the class specific portion of the  
841 WBEM server.  
842 The message sources information is a recommendation only, for implementations of  
843 a WBEM server that distinguish between a common infrastructure portion (e.g.,  
844 CIMOM) and class specific portion (e.g., providers).

## 845 6.2 Common operation parameters for all operations

846 This subclause defines commonly used operation parameters for the operations. The description of the  
847 individual operations references these operation parameters as appropriate. However, not every  
848 operation uses every one of these operation parameters.

### 849 6.2.1 IncludeQualifiers

850 The *IncludeQualifiers* operation input parameter controls whether qualifier values are returned for any  
851 returned CIM element in any returned class of a class operation.

852 Support for the *IncludeQualifiers* operation parameter in a conformant WBEM protocol is mandatory.

853 If *IncludeQualifiers* is TRUE, then any returned class and any returned CIM element within each returned  
854 class shall contain qualifier values for those qualifiers that have a value different from the default value  
855 defined in the declaration of the qualifier type. Any other qualifier values should not be included.

856 NOTE In order to inspect the scope and default value of any qualifiers that are not included in the returned class, a  
857 WBEM client can use operation *EnumerateQualifierTypes* to retrieve the qualifier type declarations that exist in a  
858 namespace.

859 If *IncludeQualifiers* is FALSE, then any returned class and any returned CIM element within each returned  
860 class shall not contain any qualifier values.

### 861 6.2.2 <element>List

862 The operation output parameters *InstanceList*, *InstancePathList*, *ClassList*, *ClassPathList*, and  
863 *QualifierTypeList* contain a sequence of elements, and are referred to as the *result set* of the operation.

864 The sequence is ordered in the sense that there is a relation of "before" and "after" between elements in  
865 the sequence and the sequence has a beginning and an end. However, this does not imply that the  
866 sequence is sorted according to some criteria.

867 Clause 5.8 defines rules for dealing with duplicate objects or duplicate object paths in the result set of an  
868 operation.

## 869 6.3 Instance operations

870 This subclause defines server operations that target a single instance, or create an instance.

### 871 6.3.1 GetInstance

#### 872 Purpose:

873 Retrieves an instance.

874 **Operation Input Parameters:**

875

Generic Name	Generic Type	Requirement	Description
InstancePath	InstancePath	Mandatory	Instance path of the instance to be retrieved (Context Parameter)
IncludedProperties	PropertyName [ ]	Optional	NULL, or unordered set of property names, acting as a restricting filter on the properties included in the returned instance

876

877 **Operation Output Parameters:**

878

Generic Name	Generic Type	Requirement	Description
Instance	InstanceSpecification	Mandatory	Representation of the retrieved instance

879

880 **Description:**

881 The *GetInstance* operation retrieves a representation of the instance referenced by *InstancePath*.

882 As defined in the description of the *InstancePath* type, the instance path of the instance to be  
 883 retrieved is interpreted in a non-polymorphic way, i.e., it references the specified instance only and  
 884 does not include any instances with the same key values in subclasses.

885 The set of properties to be included in the retrieved instance shall be determined using the following  
 886 algorithm:

- 887 • Initially, the set of properties to be included is the set of properties exposed by the creation  
 888 class of the instance. This includes all the duplicates of any duplicate non-overridden  
 889 properties.
- 890 • If the *IncludedProperties* operation input parameter is supported by the WBEM protocol  
 891 and if its value is not NULL, it acts as a restricting filter on the properties to be included in  
 892 the returned instance representation such that any properties exposed by the creation  
 893 class of the instance that are not named in that operation parameter are removed from the  
 894 set of properties to be included. Any duplicate or invalid property names in the  
 895 *IncludedProperties* operation input parameter shall be ignored. A non-NULL empty  
 896 *IncludedProperties* list removes all properties from the set of properties to be included.
- 897 • Conformant WBEM protocols may specify rules that cause properties with a value of NULL  
 898 to be removed from the set of properties to be included.

899 **Preconditions:**

- 900 • The instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail,  
 901 indicating WIPG0213.
- 902 • The creation class of the instance referenced by *InstancePath* shall exist. If it does not exist, the  
 903 operation shall fail, indicating WIPG0214.
- 904 • The namespace of the instance referenced by *InstancePath* shall exist. If it does not exist, the  
 905 operation shall fail, indicating WIPG0204.

906 **Postconditions:**

- 907       • The instance representation shall have been returned with the properties as defined in the  
908       Description paragraph for this operation.
- 909       • Requirements on ACID properties:
- 910       – Atomicity: N/A
- 911       – Update Consistency: N/A
- 912       – Isolation: Required
- 913       – Durability: N/A

914 **Error messages:**

915

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0213	Instance not found	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

916

917 **6.3.2 DeleteInstance**918 **Purpose:**

919       Deletes an instance.

920 **Operation Input Parameters:**

921

Generic Name	Generic Type	Requirement	Description
InstancePath	InstancePath	Mandatory	Instance path of the instance to be deleted (Context Parameter)

922

923 **Operation Output Parameters:**

924 None.

925 **Description:**

926 The *DeleteInstance* operation deletes the instance referenced by *InstancePath*.

927 The existence of other instances may depend on the instance to be deleted. There are multiple types  
 928 of dependent instances, and multiple options to handle such dependent instances, as defined in  
 929 5.8.9.

930 **NOTE** Any dependent instances that are deleted may reside in a different namespace (which may reside in a  
 931 different WBEM server) than the instance referenced by *InstancePath*.

932 In case of error, the consistency requirements defined in [DSP0004](#) cannot be guaranteed, but should  
 933 be attempted to be satisfied in a best effort approach. Such an approach may be to delete non-  
 934 dependent instances first. In case of error, only a subset of the instances to be deleted may have  
 935 been deleted, but each instance shall have either been deleted completely or not at all.

936 The effects of the deletion of any instances on managed resources shall be defined elsewhere. For  
 937 example, a management profile may define that the lifecycle of the instance is coupled with the  
 938 lifecycle of some underlying managed resource, and that this resource shall be deleted when the  
 939 instance is deleted.

940 **Preconditions:**

- 941 • The instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail,  
 942 indicating WIPG0213.
- 943 • The creation class of the instance referenced by *InstancePath* shall exist. If it does not exist, the  
 944 operation shall fail, indicating WIPG0214.
- 945 • The namespace of the instance referenced by *InstancePath* shall exist. If it does not exist, the  
 946 operation shall fail, indicating WIPG0204.

947 **Postconditions:**

- 948 • The instance referenced by *InstancePath* shall have been deleted.
- 949 • Any implicit deletions of dependent instances shall have happened, as defined in 5.8.9.
- 950 • Any effects of the deletion of all of these instances on any managed resources shall have  
 951 happened.
- 952 • The consistency requirements defined in [DSP0004](#) shall be satisfied for any instances related to  
 953 the deleted instances.

- 954       • Requirements on ACID properties:
- 955       – Atomicity: Required, if dependent instances are handled by rejection, as defined in 5.8.9.  
956       Recommended, if dependent instances are handled by delete propagation, as defined in  
957       5.8.9.
- 958       – Update Consistency: Required, if dependent instances are handled by rejection, as defined  
959       in 5.8.9. Recommended, if dependent instances are handled by delete propagation, as  
960       defined in 5.8.9.
- 961       – Isolation: Required, if dependent instances are handled by rejection, as defined in 5.8.9.  
962       Recommended, if dependent instances are handled by delete propagation, as defined in  
963       5.8.9.
- 964       – Durability: Required.

965 **Error messages:**

966

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0213	Instance not found	Mandatory	Class implem.	
WIPG0246	Instance cannot be deleted due to referencing association	Optional	Class implem.	
WIPG0247	Instance cannot be deleted due to multiplicity underflow	Optional	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

967

968 **6.3.3 ModifyInstance**

969 **Purpose:**

970 Changes property values of a given instance.

971 **Operation Input Parameters:**

972

Generic Name	Generic Type	Requirement	Description
InstancePath	InstancePath	Mandatory	Instance path of the instance to be modified (Context Parameter)
ModifiedInstance	InstanceSpecification	Mandatory	Representation of the modified instance, specifying the new property values
IncludedProperties	PropertyName [ ]	Optional	NULL, or unordered set of property names, acting as a restricting filter on the properties to be modified

973

974 **Operation Output Parameters:**

975 None.

976 **Description:**

977 The *ModifyInstance* operation changes property values of the instance referenced by *InstancePath*.

978 The set of properties to be changed shall be determined using the following algorithm:

- 979 • Initially, the set of properties to be changed is the set of properties specified in  
980 *ModifiedInstance*.
- 981 • If the *IncludedProperties* operation input parameter is supported by the WBEM protocol  
982 and if its value is not NULL, it acts as a restricting filter on the properties to be changed  
983 such that any properties exposed by the creation class of the instance that are not named  
984 in that operation parameter are removed from the set of properties to be changed. Any  
985 duplicate or invalid property names in the *IncludedProperties* operation input parameter  
986 shall be ignored. A non-NULL empty *IncludedProperties* list removes all properties from  
987 that set.
- 988 • Any key properties and non-modifiable properties are removed from the set of properties to  
989 be changed. As a result, specifying such properties in *ModifiedInstance* or  
990 *IncludedProperties* does not cause an error.

991 NOTE The modifiability of properties can be defined in the schema and in management profiles.

992 Conformant WBEM protocols may restrict *ModifiedInstance* to specify all properties exposed by the  
993 creation class of the instance referenced by *InstancePath*.

994 **Preconditions:**

- 995 • The instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail,  
996 indicating WIPG0213.
- 997 • The creation class of the instance referenced by *InstancePath* shall exist. If it does not exist, the  
998 operation shall fail, indicating WIPG0214.



- 999           • The namespace of the instance referenced by *InstancePath* shall exist. If it does not exist, the  
1000 operation shall fail, indicating WIPG0204.
- 1001           • The creation class of *ModifiedInstance* shall be the creation class of the instance referenced by  
1002 *InstancePath* or a superclass of that class. If this is not satisfied, the operation shall fail,  
1003 indicating WIPG0208.
- 1004           • Any properties specified in *ModifiedInstance* shall be from the set of properties exposed by the  
1005 creation class of *ModifiedInstance*. If this is not satisfied, the operation shall fail, indicating  
1006 WIPG0208.

1007 **Postconditions:**

- 1008           • The values of the properties shall have been modified as defined in the Description paragraph  
1009 for this operation.
- 1010           • The values of key properties and non-modifiable properties shall not have been modified.
- 1011           • Other properties may have changed as a result of side effects of changing properties, behavior  
1012 defined in referencing specifications, or volatility of properties.
- 1013           • The consistency requirements defined in [DSP0004](#) shall be satisfied for the modified instance.
- 1014           • Requirements on ACID properties:
- 1015           – Atomicity: Required
- 1016           – Update Consistency: Required
- 1017           – Isolation: Required
- 1018           – Durability: Required

1019 **Error messages:**

1020

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0213	Instance not found	Mandatory	Class implem.	
WIPG0220	No such property	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1021

1022 **6.3.4 CreateInstance**

1023 **Purpose:**

1024       Creates an instance of a given class.

1025 **Operation Input Parameters:**

1026

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the creation class of the instance to be created (Context Parameter)
NewInstance	InstanceSpecification	Optional	Instance representation specifying the initial property values for the instance to be created

1027

1028 **Operation Output Parameters:**

1029

Generic Name	Generic Type	Requirement	Description
InstancePath	InstancePath	Mandatory	Instance path of the new instance

1030

1031 **Description:**

1032       The *CreateInstance* operation creates an instance of the creation class referenced by *ClassPath* in  
1033       the same namespace as that creation class and returns the instance path of the new instance.

1034       The creation class is interpreted in a non-polymorphic way; that is, the creation class of the newly  
1035       created instance shall be specified creation class (and not a subclass thereof).

1036       The newly created instance shall have all properties exposed by the creation class referenced by  
1037       *ClassPath*.

- 1038 For each property, its initial value in the new instance shall be determined as follows:
- 1039 • If the *NewInstance* operation input parameter is supported, and if the property is included  
1040 in *NewInstance*, its value is used as the initial value. That is also the case if that value is  
1041 NULL.
  - 1042 • Else, if an initialization constraint is defined for the property (that is, through the class-  
1043 defined property default value, a use of the PropertyConstraint qualifier, or by a  
1044 management profile), a value satisfying that constraint is used as the initial value.
  - 1045 • Else, the initial value is implementation-defined.
- 1046 Key properties and non-writeable properties included in *NewInstance* shall be treated like any other  
1047 properties; the creation of an instance does not have the restrictions a subsequent modification has.
- 1048 Volatile properties may change their values immediately after the instance has been created.
- 1049 Instance creation based upon input data other than initial property values can be done using CIM  
1050 methods. For example, creation of an instance of *CIM\_ComputerSystem* representing a virtual  
1051 computer system could be done using a *CreateVirtualComputerSystem( )* method taking a higher-  
1052 level specification of the virtual computer system as input.
- 1053 Other instances may come into existence implicitly during the course of processing the  
1054 *CreateInstance* operation. As defined in [DSP1001](#), management profiles may specify the rules for  
1055 such implicitly created instances.
- 1056 Any such implicitly created instances may reside in the same or a different namespace (which may  
1057 reside in a different WBEM server) than the namespace of the creation class referenced by  
1058 *ClassPath*.
- 1059 In case of error, the consistency requirements defined in [DSP0004](#) should be attempted to be  
1060 satisfied in a best effort approach. In case of error, only a subset of the instances to be created may  
1061 have been created, but each instance shall have either been created completely or not at all.
- 1062 As defined in [DSP1001](#), management profiles may specify the effects of the creation of instances on  
1063 managed resources. For example, a management profile may define that the lifecycle of the instance  
1064 is coupled with the lifecycle of some underlying managed resource, and that this resource shall be  
1065 created when the instance is created.
- 1066 **Preconditions:**
- 1067 • The instance to be created shall not exist in the namespace specified by *ClassPath*. If this is not  
1068 satisfied, the operation shall fail, indicating WIPG0216.
  - 1069 • The class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail,  
1070 indicating WIPG0214.
  - 1071 • The namespace of the class referenced by *ClassPath* shall exist. If it does not exist, the  
1072 operation shall fail, indicating WIPG0204.
  - 1073 • The creation class of *NewInstance* shall be the class referenced by *ClassPath* or a superclass  
1074 of that class. If this is not satisfied, the operation shall fail, indicating WIPG0208.
  - 1075 • Any properties specified in *NewInstance* shall be from the set of properties exposed by the  
1076 class referenced by *ClassPath*. If this is not satisfied, the operation shall fail, indicating  
1077 WIPG0208.

- 1078 • If the schema definition of the class referenced by *ClassPath* or any implemented management
- 1079 profiles require that *NewInstance* includes a property, but that property is not included in
- 1080 *NewInstance*, the operation shall fail, indicating WIPG0249.
  
- 1081 • If the schema definition of the class referenced by *ClassPath* or any implemented management
- 1082 profiles require that *NewInstance* does not include a property, but that property is included in
- 1083 *NewInstance*, the operation shall fail, indicating WIPG0249.

1084 **Postconditions:**

- 1085 • The instance shall have been created as defined in the Description paragraph for this operation.
- 1086 • Any management profile defined implicit creations of other instances shall have happened.
- 1087 • Any management profile defined effects of the creation of all of these instances on any
- 1088 managed resources shall have happened.
- 1089 • Requirements on ACID properties:
  - 1090 – Atomicity: Required
  - 1091 – Update Consistency: Required
  - 1092 – Isolation: Required
  - 1093 – Durability: Required

1094 **Error messages:**

1095

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0216	Instance already exists	Mandatory	Class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1096

## 1097 **6.4 Instance enumeration operations**

1098 This subclause defines server operations that enumerate instances and return their representations and  
1099 instance paths by means of subsequent pull operations.

1100 The common pattern for these operations is that an enumeration session gets established through an  
1101 "Open" operation, also establishing the kind of operation and the kind of items to be returned (instance  
1102 representations together with instance paths, or just instance paths), and subsequent repeated  
1103 executions of a "Pull" operation on the enumeration session are used to retrieve the items. Optionally, the  
1104 "Open" operation can also pull a first set of items.

1105 The pulled instance enumeration operations consist of the following individual operations:

1106 • Open operations:

1107 OpenEnumerateInstances – Open an enumeration of instances of a given class for returning  
1108 their representations and instance paths

1109 OpenAssociators – Open an enumeration of instances associated to a given source instance for  
1110 returning their representations and instance paths

1111 OpenReferences – Open an enumeration of association instances referencing a given source  
1112 instance for returning their representations and instance paths

1113 OpenQueryInstances – Open an enumeration of instances representing a query result for  
1114 returning only their instance representations

1115 • Pull operations:

1116 PullInstancesWithPath – Pull operation for retrieving instance representations with instance  
1117 paths

1118 PullInstances – Pull operation for retrieving instance representations (without instance paths),  
1119 representing query results

1120 • Other operations:

1121 CloseEnumeration – Close an open enumeration

### 1122 **6.4.1 General behavioral rules**

1123 A central concept of the pulled instance enumeration operations is the "enumeration session". An  
1124 enumeration session can be thought of as a context in which the operations perform their work, and  
1125 which determines the set of instances to be enumerated. In order to process the operations related to an  
1126 enumeration session, some of the operation parameters of the Open operation need to be maintained as  
1127 long as the enumeration session is open, as well as some state data about where the enumeration  
1128 session is with respect to instances already returned.

1129 From a WBEM client's perspective, an enumeration session is represented as an enumeration context  
1130 value. A successful Open operation establishes the enumeration session and returns an enumeration  
1131 context value representing the open enumeration session. The enumeration context value is used as an

1132 operation input/output parameter in subsequent Pull operations on that enumeration session. The  
1133 enumeration context value shall uniquely identify the open enumeration session within the target  
1134 namespace of the Open operation that established the enumeration session. This does not require the  
1135 enumeration context value to be time-unique, i.e., it may be reused for a new enumeration session after  
1136 the old enumeration session was closed. It is valid for a WBEM server to use NULL as an enumeration  
1137 context value representing a closed enumeration session, but a WBEM client shall not rely on that to  
1138 detect that an enumeration session has been closed.

1139 Defining the enumeration context value in Pull operations not only as an operation input parameter but  
1140 also as an operation output parameter allows the WBEM server to change the enumeration context value  
1141 during the execution of a Pull operation. This allows for different implementation approaches for the  
1142 WBEM server, which are transparent for the WBEM client.

1143 Example approaches are:

- 1144 • maintaining any state data describing the enumeration session internally in the WBEM server.  
1145 In this approach, the enumeration context value does not need to change in subsequent Pull  
1146 operations. It is used by the WBEM server only to identify the internal state data for the open  
1147 enumeration session, but it is not used to store any of the state data in it. A variation of this  
1148 approach is to hand back modified enumeration context values for additional WBEM server side  
1149 sequence checking.
- 1150 • maintaining any state data describing the enumeration session on the WBEM client side only. In  
1151 this approach, all state data is stored in the enumeration context value, and the WBEM server  
1152 does not maintain any state data about the enumeration session, essentially being completely  
1153 stateless with respect to the enumeration session.
- 1154 • a combination of the two previous approaches

1155 A WBEM server may support keeping enumeration sessions open across connection terminations and  
1156 shutdowns of the server. Objects may be created, deleted or modified concurrently with an enumeration  
1157 session that involves these objects. Such changes may or may not be reflected in the enumeration set.  
1158 Therefore, there is no guarantee to the WBEM client that the enumeration set represents a consistent  
1159 snapshot of its objects at a point in time. However, the WBEM server should make a best effort attempt  
1160 for the returned enumeration set to represent a consistent snapshot of its objects at a point in time. The  
1161 order of objects in the enumeration set is undefined.

1162 This specification does not define any restrictions on the number of enumeration sessions that can be  
1163 established or executed on concurrently in the same WBEM server or by the same WBEM client. This  
1164 remains true even if the enumeration sets of such concurrently established enumeration sessions contain  
1165 the same objects.

1166 With the exception of the CloseEnumeration operation, all operations on a particular enumeration session  
1167 shall be executed sequentially. An enumeration session can be open or closed. The enumeration session  
1168 is considered open if operations using its enumeration context value as an operation input parameter can  
1169 be executed successfully. It is opened by the successful completion of an Open operation and closed by  
1170 one of the following:

- 1171 • Successful completion of a CloseEnumeration operation
- 1172 • Successful completion of an Open or Pull operation that has its *EndOfSequence* operation  
1173 output parameter set to TRUE. In other words, reaching the end of the enumeration set closes  
1174 the enumeration session implicitly
- 1175 • Unsuccessful completion of a Pull operation when *ContinueOnError* had not been requested
- 1176 • WBEM server side decision to close the enumeration session based upon an operation timeout
- 1177 • WBEM server side decision to close an enumeration session during an operation on that  
1178 enumeration session based upon exceeding server limits

1179 A conformant WBEM server may support closure of enumeration sessions based upon exceeding server  
 1180 limits. Potential examples for such a decision may be Pull operations with no objects requested that are  
 1181 repeated with a high frequency on the same enumeration session. If a WBEM server supports closure of  
 1182 enumeration sessions based upon exceeding server limits, it shall make the decision to close an  
 1183 enumeration session during an operation on that enumeration session. (There is no way to indicate the  
 1184 reason for the closure if the decision is made elsewhere.)

## 1185 **6.4.2 Common operation parameters for the open operations**

1186 This subclause defines commonly used operation parameters for the Open operations. The description of  
 1187 the individual Open operations references these operation parameters as appropriate. However, not  
 1188 every Open operation uses every one of these common operation parameters.

### 1189 **6.4.2.1 EnumerationContext**

1190 The *EnumerationContext* operation output parameter is the enumeration context value representing the  
 1191 enumeration session. See 6.4.1 for a definition of the concepts of *enumeration session* and *enumeration*  
 1192 *context value*.

### 1193 **6.4.2.2 EndOfSequence**

1194 NOTE This operation output parameter is also used for Pull operations.

1195 The *EndOfSequence* operation output parameter indicates whether the enumeration session is  
 1196 exhausted.

1197 If *EndOfSequence* is TRUE upon successful completion of an operation, no more objects are available  
 1198 and the WBEM server shall have closed the enumeration session, releasing any possibly allocated  
 1199 compute resources related to the enumeration session.

1200 If the returned enumeration set is empty, it is valid for a WBEM server to set *EndOfSequence* to TRUE,  
 1201 even if *MaxObjectCount* was 0. In this case, the enumeration session will be closed upon successful  
 1202 completion of the operation.

1203 If *EndOfSequence* is FALSE upon successful completion of an operation, there may be additional  
 1204 elements available and the WBEM server shall not have closed the enumeration session.

### 1205 **6.4.2.3 FilterQueryLanguage and FilterQueryString**

1206 The *FilterQueryLanguage* and *FilterQueryString* operation input parameters define a filter query that acts  
 1207 as an additional restricting filter on the set of instances about which information is returned.

1208 Support for the *FilterQueryLanguage* and *FilterQueryString* operation parameters is conditional on  
 1209 support in the WBEM protocol for filter queries in pulled instance enumeration operations.

1210 If the WBEM protocol supports filter queries in pulled instance enumeration operations, the following rules  
 1211 apply:

- 1212 • Conformant WBEM protocols shall require that the DMTF Filter Query Language (FQL) defined  
 1213 in [DSP0212](#) is supported for the filter queries. Conformant WBEM protocols may support  
 1214 additional filter query languages.
- 1215 • If *FilterQueryLanguage* is not NULL, additional filtering is requested and the following rules  
 1216 apply:
  - 1217 – *FilterQueryLanguage* shall specify a valid query language and *FilterQueryString* shall  
 1218 be a valid query in that query language. Neither the query language nor the format of  
 1219 the filter query is defined by this specification. Conformant WBEM protocols shall

- 1220 define a mechanism whereby WBEM servers can declare the set of query languages  
1221 that are valid for *FilterQueryLanguage*.
- 1222 – A filter query may specify any result set (e.g., SELECT list), but because the purpose  
1223 of the filter query is to restrict the set of instances about which information is returned,  
1224 its result set shall be ignored. The filter query shall not define any ordering criteria.  
1225 The filter query shall not define any grouping of objects. Operations using filter queries  
1226 may specify additional constraints on the filter query.
  - 1227 – If the WBEM server infrastructure does not support filtered enumerations, the WBEM  
1228 server shall return failure with message WIPG0237 (Filter queries not supported by  
1229 WBEM server infrastructure).
  - 1230 – If the CIM class implementation does not support filtered enumerations, the WBEM  
1231 server shall return failure with message WIPG0244 (Filter queries not supported by  
1232 class implementation).
  - 1233 • If *FilterQueryLanguage* is NULL, no additional filtering shall take place, and *FilterQueryString*  
1234 shall be NULL.
  - 1235 – If *FilterQueryString* is not NULL, the WBEM server shall return failure with message  
1236 WIPG0208 (Invalid operation input parameter value).
- 1237 If the WBEM protocol does not support filter queries in pulled instance enumeration operations, no  
1238 additional filtering shall take place.

#### 1239 6.4.2.4 OperationTimeout

- 1240 The *OperationTimeout* operation input parameter determines the "operation timeout". The operation  
1241 timeout is the minimum time the WBEM server shall maintain the open enumeration session after the last  
1242 Open or Pull operation (unless the enumeration session was closed during that last operation). If the  
1243 operation timeout is exceeded, the WBEM server may close the enumeration session at any time,  
1244 releasing any possibly allocated compute resources related to the enumeration session.
- 1245 Support for the *OperationTimeout* operation parameter in a conformant WBEM protocol is mandatory.
- 1246 An *OperationTimeout* of 0 means that there is no operation timeout, i.e., the enumeration session is never  
1247 closed based on time.
- 1248 If *OperationTimeout* is NULL, the WBEM server shall choose an operation timeout.
- 1249 All other values for *OperationTimeout* specify the operation timeout in seconds.
- 1250 A WBEM server may restrict the set of allowable values for *OperationTimeout*. This specifically includes  
1251 the possibility for the WBEM server to not allow 0 (no timeout). If the specified value is not an allowable  
1252 value, the WBEM server shall return failure with error message WIPG0242 (Invalid timeout). Conformant  
1253 WBEM protocols shall define a mechanism whereby WBEM servers can declare the allowable values for  
1254 *OperationTimeout*.

#### 1255 6.4.2.5 ContinueOnError

- 1256 The *ContinueOnError* operation input parameter, if TRUE, requests continuation on error. Continuation on  
1257 error is the ability to resume an enumeration session successfully after a Pull operation that returned an  
1258 error. A conformant WBEM server may support continuation on error. Conformant WBEM protocols shall  
1259 define a mechanism whereby WBEM servers can declare support for continuation on error.
- 1260 Support for the *ContinueOnError* operation parameter is conditional on support in the WBEM protocol for  
1261 client side control of continuation on error for pulled instance enumeration operations.



1262 If the WBEM protocol supports client side control of continuation on error for pulled instance enumeration  
 1263 operations, the following rules apply:

- 1264 • If a WBEM server does not support continuation on error and if *ContinueOnError* is TRUE, it  
 1265 shall return failure with error message WIPG0235 (Continuation on error not supported).
- 1266 • If a WBEM server supports continuation on error, it shall support it as follows: If  
 1267 *ContinueOnError* is TRUE, the enumeration session shall remain open when a Pull operation  
 1268 returns failure, and any subsequent successful Pull operations shall return the set of elements  
 1269 that would have been returned if the failing Pull operations had been successful, subject to the  
 1270 consistency rules defined in 5.8. If *ContinueOnError* is FALSE, the enumeration session shall  
 1271 be closed when a Pull operation returns failure.

1272 If the WBEM protocol does not support client side control of continuation on error for pulled instance  
 1273 enumeration operations, it shall define requirements for the behavior of the WBEM server with respect to  
 1274 continuation on error.

1275 **6.4.2.6 MaxObjectCount**

1276 NOTE This operation output parameter is also used for Pull operations.

1277 The *MaxObjectCount* operation input parameter defines the maximum number of objects that may be  
 1278 returned by this operation. Any uint32 number is valid, including 0. The WBEM server may deliver any  
 1279 number of objects up to *MaxObjectCount* but shall not deliver more than *MaxObjectCount* objects.

1280 Support for the *MaxObjectCount* operation parameter in a conformant WBEM protocol is mandatory.

1281 A conformant WBEM server implementation may choose to never return any elements during an  
 1282 operation, regardless of the value of *MaxObjectCount*.

1283 A WBEM client may use a *MaxObjectCount* value of 0 to specify that it does not want to retrieve any  
 1284 instances in the operation.

1285 **6.4.3 OpenEnumerateInstances**

1286 **Purpose:**

1287 Establish and open an enumeration session for enumerating the instances of a given class and  
 1288 optionally return a first set of their instance representations and instance paths.

1289 **Operation Input Parameters:**

1290

Generic Name	Generic Type	Requirement	Description
EnumClassPath	ClassPath	Mandatory	Class path of the class whose instances are to be enumerated (Context Parameter)
FilterQueryString	QueryString	Conditional	NULL, or query string of a filter query that is acting as an additional restricting filter on the set of instances to be enumerated, as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.

Generic Name	Generic Type	Requirement	Description
FilterQueryLanguage	QueryLanguage	Conditional	NULL, or query language of the filter query specified in <i>FilterQueryString</i> , as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.
IncludedProperties	PropertyName [ ]	Optional	NULL, or unordered set of property names to be included, acting as a restricting filter on the properties included in the returned instance representations
OperationTimeout	uint32	Mandatory	Operation timeout, as defined in 6.4.2.4
ContinueOnError	boolean	Conditional	Indicates whether the enumeration session should be continued in case of error, as defined in 6.4.2.5 Condition: WBEM protocol supports client side control of continuation on error for pulled instance enumeration operations.
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.2.6

1291

1292 **Operation Output Parameters:**

1293

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecificationWithPath [ ]	Mandatory	Sequence of the returned first set of instance representations and instance paths
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.2.1
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.2.2

1294

1295 **Description:**

1296 The *OpenEnumerateInstances* operation establishes and opens an enumeration session for  
 1297 enumerating all instances of the class referenced by *EnumClassPath*, including instances of any of  
 1298 its subclasses. That enumeration session allows retrieving the instance representations and instance  
 1299 paths of these instances through successive *PullInstancesWithPath* operations (see 6.4.8). Retrieval  
 1300 of a first set of instance representations and instance paths may be requested by setting  
 1301 *MaxObjectCount* to a value > 0.

1302 The set of instances to be enumerated throughout the entire enumeration session shall be  
 1303 determined using the following algorithm:

- 1304 • Initially, the set of instances to be enumerated is the set of instances in the namespace of  
 1305 the class referenced by *EnumClassPath*, whose creation class is the class referenced by  
 1306 *EnumClassPath* or a subclass of that class.

- 1307           • If the WBEM protocol supports filter queries for pulled instance enumeration operations  
 1308           (that is, the *FilterQueryString* and *FilterQueryLanguage* operation parameters) and  
 1309           *FilterQueryLanguage* is not NULL, *FilterQueryString* acts as a restricting filter on the  
 1310           instances to be enumerated such that any instances not selected by the filter query for its  
 1311           result set are removed from the set of instances. The filter query shall query only the class  
 1312           referenced by *EnumClassPath*. See also 6.4.2.3.

1313           The set of instances to be enumerated throughout the entire enumeration session should not contain  
 1314           any duplicate instances, as defined in 5.8.4. Because instances to be enumerated all exist in the  
 1315           same namespace, a determination of duplicate instances (for example by a WBEM client) can be  
 1316           done on the basis of their model paths only.

1317           The set of instances to be returned (as instance representations and instance paths) is the first set of  
 1318           instances from the set of instances to be enumerated throughout the entire enumeration session,  
 1319           such that no more than *MaxObjectCount* instances are returned. Returning no instances does not  
 1320           imply that the enumeration session has been exhausted. Only the *EndOfSequence* operation output  
 1321           parameter indicates whether the enumeration session has been exhausted.

1322           The set of properties to be included in any returned instance representations shall be determined  
 1323           using the following algorithm:

- 1324           • Initially, the set of properties to be included is the set of properties exposed by the creation  
 1325           class of the instance. This includes all the duplicates of any duplicate non-overridden  
 1326           properties.
- 1327           • If the *IncludedProperties* operation input parameter is supported by the WBEM protocol  
 1328           and if its value is not NULL, it acts as a restricting filter on the properties to be included in  
 1329           the returned instance representations such that any properties exposed by the creation  
 1330           class of the instance that are not named in that operation parameter are removed from the  
 1331           set of properties to be included. Any duplicate or invalid property names in the  
 1332           *IncludedProperties* operation input parameter shall be ignored. A non-NULL empty  
 1333           *IncludedProperties* list removes all properties from the set of properties to be included.
- 1334           • Conformant WBEM protocols may specify rules that cause properties with a value of NULL  
 1335           to be removed from the set of properties to be included.

#### 1336 **Preconditions:**

- 1337           • The class referenced by *EnumClassPath* shall exist. If it does not exist, the operation shall fail,  
 1338           indicating WIPG0214.
- 1339           • The namespace of the class referenced by *EnumClassPath* shall exist. If it does not exist, the  
 1340           operation shall fail, indicating WIPG0204.
- 1341           • If a filter query is specified,
  - 1342           – the query language specified in the *FilterQueryLanguage* operation parameter shall be  
 1343           valid. If this is not satisfied, the operation shall fail, indicating WIPG0221.
  - 1344           – the query specified in the *FilterQueryString* operation parameter shall be a valid query in  
 1345           the query language specified in the *FilterQueryLanguage* operation parameter. If this is not  
 1346           satisfied, the operation shall fail, indicating WIPG0222 or WIPG0223.

#### 1347 **Postconditions:**

- 1348           • The enumeration session shall have been established and opened.
- 1349           • A first set of instance representations and instance paths shall have been returned as described  
 1350           in the Description paragraph for this operation.

- 1351 • Requirements on ACID properties:
  - 1352 – Atomicity: Required (related to the creation of an enumeration context that is maintained by
  - 1353 the WBEM server)
  - 1354 – Update Consistency: N/A
  - 1355 – Isolation: Required at the level of single instances, as defined in 5.8.
  - 1356 – Durability: Required (related to creation of an enumeration context that is maintained by
  - 1357 the WBEM server)

**Error messages:**

1358  
1359

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0242	Invalid timeout	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0235	Continuation on error not supported	Mandatory	Infrastructure, class implem.	
WIPG0237	Filter queries not supported by WBEM server infrastructure	Optional	Infrastructure	
WIPG0244	Filter queries not supported by class implementation	Optional	Class implem.	
WIPG0221	Unknown query language	Mandatory	Infrastructure, class implem.	
WIPG0222	Query language feature not supported	Mandatory	Infrastructure, class implem.	
WIPG0223	Invalid query	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1360

1361 **6.4.4 OpenAssociators**1362 **Purpose:**

1363 Establish and open an enumeration session for enumerating the instances that are associated with a  
 1364 given source instance and optionally return a first set of their instance representations and instance  
 1365 paths.

1366 **Operation Input Parameters:**

1367

Generic Name	Generic Type	Requirement	Description
SourceInstancePath	InstancePath	Mandatory	Instance path of the source instance (Context Parameter)
AssociationClassName	ClassName	Mandatory	NULL, or name of the association class, acting as a restricting filter on the returned instances
AssociatedClassName	ClassName	Mandatory	NULL, or name of the associated class on any far end of the association, acting as a restricting filter on the returned instances
SourceRoleName	PropertyName	Mandatory	NULL, or name of the role on the source end of the association, acting as a restricting filter on the returned instances
AssociatedRoleName	PropertyName	Mandatory	NULL, or name of the role on any far end of the association, acting as a restricting filter on the returned instances
FilterQueryString	QueryString	Conditional	NULL, or query string of a filter query that is acting as an additional restricting filter on the set of returned instances, as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.
FilterQueryLanguage	QueryLanguage	Conditional	NULL, or query language of the filter query specified in <i>FilterQueryString</i> , as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.
IncludedProperties	PropertyName [ ]	Optional	NULL, or unordered set of property names to be included, acting as a restricting filter on the properties included in the returned instances
OperationTimeout	uint32	Mandatory	Operation timeout, as defined in 6.4.2.4

Generic Name	Generic Type	Requirement	Description
ContinueOnError	boolean	Conditional	Indicates whether the enumeration session should be continued in case of error, as defined in 6.4.2.5 Condition: WBEM protocol supports client side control of continuation on error for pulled instance enumeration operations.
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.2.6

1368

1369 **Operation Output Parameters:**

1370

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecificationWithPath [ ]	Mandatory	Sequence of the returned first set of instance representations and instance paths
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.2.1
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.2.2

1371

1372 **Description:**

1373 The *OpenAssociators* operation establishes and opens an enumeration session for enumerating  
 1374 instances that are associated with the specified source instance. That enumeration session allows  
 1375 retrieving the instance representations and instance paths of these instances through successive  
 1376 *PullInstancesWithPath* operations (see 6.4.8). Retrieval of a first set of those instances together with  
 1377 their instance paths may be requested by setting *MaxObjectCount* to a value > 0.

1378 The set of instances to be enumerated throughout the entire enumeration session shall be  
 1379 determined using the following algorithm:

- 1380 • Initially, the set of instances to be enumerated is the set of all instances associated to the  
 1381 source instance referenced by *SourceInstancePath*. These associations may be instances  
 1382 of different association classes. If the source instance does not exist, the operation shall  
 1383 succeed with an empty result set (even when its creation class does not exist). However, if  
 1384 the namespace of the source instance does not exist, the operation shall fail, indicating  
 1385 WIPG0204.

1386 The result set should not contain any duplicate instances, as defined in 5.8.4. However,  
 1387 different far ends may reference the same instance, and in such cases, the instance shall  
 1388 be contained in the result set once for each such reference.

- 1389 • If the *AssociationClassName* operation input parameter is not NULL, it acts as a restricting  
 1390 filter on the instances to be enumerated such that each instance that is associated with the  
 1391 source instance using an association whose creation class or one of its superclasses does  
 1392 not have the name specified in *AssociationClassName*, is removed from the set of  
 1393 instances to be enumerated. There shall be no validity checking performed for the  
 1394 *AssociationClassName* operation input parameter; if the specified class does not exist, the  
 1395 operation shall succeed with an empty result (because the filter did not match).

- 1396
- 1397
- 1398
- 1399
- 1400
- 1401
- 1402
- If the *AssociatedClassName* operation input parameter is not NULL, it acts as a restricting filter on the instances to be enumerated such that each instance whose creation class or one of its superclasses does not have the name specified in *AssociatedClassName*, is removed from the set of instances to be enumerated. There shall be no validity checking performed for the *AssociatedClassName* operation input parameter; if the specified class does not exist, the operation shall succeed with an empty result (because the filter did not match).
- 1403
- 1404
- NOTE Specifying a non-NULL value for *AssociatedClassName* ensures that the returned instances have the class specified in *AssociatedClassName* as a common superclass.
- 1405
- If the *SourceRoleName* operation input parameter is not NULL, it acts as a restricting filter on the instances to be enumerated such that each instance that is associated with the source instance using an association class that has a role name on the source end that is not the role name specified in *SourceRoleName*, is removed from the set of instances to be enumerated. There shall be no validity checking performed for the *SourceRoleName* operation input parameter; if the specified role does not exist, the operation shall succeed with an empty result (because the filter did not match).
- 1406
- 1407
- 1408
- 1409
- 1410
- 1411
- If the *AssociatedRoleName* operation input parameter is not NULL, it acts as a restricting filter on the instances to be enumerated such that each instance that is associated with the source instance using an association class that has a role name on the end referencing that instance that is not the role name specified in *AssociatedRoleName*, is removed from the set of instances to be enumerated. There shall be no validity checking performed for the *AssociatedRoleName* operation input parameter; if the specified role does not exist, the operation shall succeed with an empty result (because the filter did not match).
- 1412
- 1413
- 1414
- 1415
- 1416
- 1417
- 1418
- If the WBEM protocol supports filter queries for pulled instance enumeration operations (that is, the *FilterQueryString* and *FilterQueryLanguage* operation parameters) and *FilterQueryLanguage* is not NULL, *FilterQueryString* acts as a restricting filter on the instances to be enumerated such that any instances not selected by the filter query for its result set are removed from the set of instances. The filter query shall query only the class specified in *AssociatedClassName* (e.g., in the CQL FROM-clause). See also 6.4.2.3.
- 1419
- 1420
- 1421
- 1422
- 1423
- 1424
- 1425
- 1426
- 1427
- 1428
- The set of instances to be enumerated throughout the entire enumeration session should not contain any duplicate instances, as defined in 5.8.4. Because the set of returned instances contains only instances that exist in the same namespace, a determination of duplicate instances can be done on the basis of their model paths only.
- 1429
- 1430
- 1431
- 1432
- 1433
- The set of instances to be returned (as instance representations and instance paths) is the first set of instances from the set of instances to be enumerated throughout the entire enumeration session, such that no more than *MaxObjectCount* instances are returned. Returning no instances does not imply that the enumeration session has been exhausted. Only the *EndOfSequence* operation output parameter indicates whether the enumeration session has been exhausted.

1434 The set of properties to be included in any returned instances shall be determined using the following  
1435 algorithm:

- 1436 • Initially, the set of properties to be included is the set of properties exposed by the creation  
1437 class of the instance. This includes all the duplicates of any duplicate non-overridden  
1438 properties.
- 1439 • If the *IncludedProperties* operation input parameter is supported by the WBEM protocol  
1440 and if its value is not NULL, it acts as a restricting filter on the properties to be included in  
1441 the returned instances such that any properties exposed by the creation class of the  
1442 instance that are not named in that operation parameter are removed from the set of  
1443 properties to be included. Any duplicate or invalid property names in the  
1444 *IncludedProperties* operation input parameter shall be ignored. A non-NULL empty  
1445 *IncludedProperties* list removes all properties from the set of properties to be included.
- 1446 • Conformant WBEM protocols may specify rules that cause properties with a value of NULL  
1447 to be removed from the set of properties to be included.

#### 1448 **Preconditions:**

- 1449 • The namespace of the source instance referenced by *SourceInstancePath* shall exist. If it does  
1450 not exist, the operation shall fail, indicating WIPG0204.
- 1451 • If a filter query is specified,
  - 1452 – the query language specified in the *FilterQueryLanguage* operation parameter shall be  
1453 valid. If this is not satisfied, the operation shall fail, indicating WIPG0221.
  - 1454 – the query specified in the *FilterQueryString* operation parameter shall be a valid query in  
1455 the query language specified in the *FilterQueryLanguage* operation parameter. If this is not  
1456 satisfied, the operation shall fail, indicating WIPG0222 or WIPG0223.
  - 1457 – the *AssociatedClassName* operation input parameter shall be non-NULL. If this is not  
1458 satisfied, the operation shall fail, indicating WIPG0208.
- 1459 • The *IncludedProperties* operation parameter, if supported by the WBEM protocol, shall only be  
1460 specified with a non-NULL value if the *AssociatedClassName* operation input parameter is also  
1461 non-NULL. If this is not satisfied, the operation shall fail, indicating WIPG0208.
- 1462 • The namespace of any returned instance paths shall exist. If it does not exist, the operation may  
1463 fail, indicating WIPG0204. Note that cross-namespace association traversals may return  
1464 instance paths in a server or namespace that is different from the server or namespace of the  
1465 source instance.
- 1466 • The creation class of any returned instance paths shall exist in their namespace. If it does not  
1467 exist, the operation may fail, indicating WIPG0214.

#### 1468 **Postconditions:**

- 1469 • The enumeration session shall have been established and opened.
- 1470 • A first set of instances with their instance paths shall have been returned as described in the  
1471 Description paragraph for this operation.
- 1472 • Requirements on ACID properties:
  - 1473 – Atomicity: Required (related to the creation of an enumeration context that is maintained by  
1474 the WBEM server)
  - 1475 – Update Consistency: N/A



- 1476 – Isolation: Required at the level of single instances, as defined in 5.8.
- 1477 – Durability: Required (related to creation of an enumeration context that is maintained by
- 1478 the WBEM server)

1479 **Error messages:**

1480

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	For input namespace
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0242	Invalid timeout	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0235	Continuation on error not supported	Mandatory	Infrastructure, class implem.	
WIPG0237	Filter queries not supported by WBEM server infrastructure	Optional	Infrastructure	
WIPG0244	Filter queries not supported by class implementation	Optional	Class implem.	
WIPG0221	Unknown query language	Mandatory	Infrastructure, class implem.	
WIPG0222	Query language feature not supported	Mandatory	Infrastructure, class implem.	
WIPG0223	Invalid query	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0204	Namespace not found	Optional	Infrastructure	For namespace of returned instance paths
WIPG0214	Class not found	Optional	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1481

1482 **6.4.5 OpenReferences**

1483 **Purpose:**

1484 Establish and open an enumeration session for enumerating the association instances that reference  
 1485 a given source instance and optionally return a first set of their instance representations and instance  
 1486 paths.

1487 **Operation Input Parameters:**

1488

Generic Name	Generic Type	Requirement	Description
SourceInstancePath	InstancePath	Mandatory	Instance path of the source instance (Context Parameter)
AssociationClassName	ClassName	Mandatory	NULL, or name of the association class, acting as a restricting filter on the returned instances
SourceRoleName	PropertyName	Mandatory	NULL, or name of the role on the source end of the association, acting as a restricting filter on the returned instances
FilterQueryString	QueryString	Conditional	NULL, or query string of a filter query that is acting as an additional restricting filter on the set of returned instances, as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.
FilterQueryLanguage	QueryLanguage	Conditional	NULL, or query language of the filter query specified in <i>FilterQueryString</i> , as defined in 6.4.2.3 Condition: WBEM protocol supports filter queries for pulled instance enumeration operations.
IncludedProperties	PropertyName [ ]	Optional	NULL, or unordered set of property names to be included, acting as a restricting filter on the properties included in the returned instances
OperationTimeout	uint32	Mandatory	Operation timeout, as defined in 6.4.2.4

Generic Name	Generic Type	Requirement	Description
ContinueOnError	boolean	Conditional	Indicates whether the enumeration session should be continued in case of error, as defined in 6.4.2.5 Condition: WBEM protocol supports client side control of continuation on error for pulled instance enumeration operations.
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.2.6

1489

1490 **Operation Output Parameters:**

1491

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecificationWithPath [ ]	Mandatory	Sequence of the returned first set of instance representations and instance paths
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.2.1
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.2.2

1492

1493 **Description:**

1494 The *OpenReferences* operation establishes and opens an enumeration session for enumerating the  
 1495 association instances that reference the specified source instance. That enumeration session allows  
 1496 retrieving the instance representations and instance paths of these instances through successive  
 1497 *PullInstancesWithPath* operations (see 6.4.8). Retrieval of a first set of those instances together with  
 1498 their instance paths may be requested by setting *MaxObjectCount* to a value > 0.

1499 The set of instances to be enumerated throughout the entire enumeration session shall be  
 1500 determined using the following algorithm:

- 1501 • Initially, the set of instances to be enumerated is the set of all instances referencing the  
 1502 source instance referenced by *SourceInstancePath*. These associations may be instances  
 1503 of different association classes. If the source instance does not exist, the operation shall  
 1504 succeed with an empty result set (even when its creation class does not exist). However, if  
 1505 the namespace of the source instance does not exist, the operation shall fail, indicating  
 1506 WIPG0204.
- 1507 • If the *AssociationClassName* operation input parameter is not NULL, it acts as a restricting  
 1508 filter on the instances to be enumerated such that each association instance whose  
 1509 creation class or one of its superclasses does not have the name specified in  
 1510 *AssociationClassName*, is removed from the set of instances to be enumerated. There  
 1511 shall be no validity checking performed for the *AssociationClassName* operation input  
 1512 parameter; if the specified class does not exist, the operation shall succeed with an empty  
 1513 result (because the filter did not match).

1514 NOTE Specifying a non-NULL value for *AssociationClassName* ensures that the returned  
 1515 instances have the class specified in *AssociationClassName* as a common superclass.

- 1516           • If the *SourceRoleName* operation input parameter is not NULL, it acts as a restricting filter  
1517 on the instances to be enumerated such that each association instance whose creation  
1518 class does not have the role name specified in *SourceRoleName* on the end referencing  
1519 the source instance, is removed from the set of instances to be enumerated. There shall be  
1520 no validity checking performed for the *SourceRoleName* operation input parameter; if the  
1521 specified role does not exist, the operation shall succeed with an empty result (because the  
1522 filter did not match).
- 1523           • If the WBEM protocol supports filter queries for pulled instance enumeration operations  
1524 (that is, the *FilterQueryString* and *FilterQueryLanguage* operation parameters) and  
1525 *FilterQueryLanguage* is not NULL, *FilterQueryString* acts as a restricting filter on the  
1526 instances to be enumerated such that any instances not selected by the filter query for its  
1527 result set are removed from the set of instances. The filter query shall query only the class  
1528 specified in *AssociationClassName* (e.g., in the CQL FROM-clause). See also 6.4.2.3.
- 1529           The set of instances to be enumerated throughout the entire enumeration session should not contain  
1530 any duplicate instances, as defined in 5.8.4. Because the set of returned instances contains only  
1531 instances that exist in the same namespace, so any determination of duplicate instances (for  
1532 example by a WBEM client) may be done on the basis of their model paths.
- 1533           The set of instances to be returned (as instance representations and instance paths) is the first set of  
1534 instances from the set of instances to be enumerated throughout the entire enumeration session,  
1535 such that no more than *MaxObjectCount* instances are returned. Returning no instances does not  
1536 imply that the enumeration session has been exhausted. Only the *EndOfSequence* operation output  
1537 parameter indicates whether the enumeration session has been exhausted.
- 1538           The set of properties to be included in any returned instances shall be determined using the following  
1539 algorithm:
- 1540           • Initially, the set of properties to be included is the set of properties exposed by the creation  
1541 class of the instance. This includes all the duplicates of any duplicate non-overridden  
1542 properties.
- 1543           • If the *IncludedProperties* operation input parameter is supported by the WBEM protocol  
1544 and if its value is not NULL, it acts as a restricting filter on the properties to be included in  
1545 the returned instances such that any properties exposed by the creation class of the  
1546 instance that are not named in that operation parameter are removed from the set of  
1547 properties to be included. Any duplicate or invalid property names in the  
1548 *IncludedProperties* operation input parameter shall be ignored. A non-NULL empty  
1549 *IncludedProperties* list removes all properties from the set of properties to be included.
- 1550           • Conformant WBEM protocols may specify rules that cause properties with a value of NULL  
1551 to be removed from the set of properties to be included.

#### 1552 **Preconditions:**

- 1553           • The namespace of the source instance referenced by *SourceInstancePath* shall exist. If it does  
1554 not exist, the operation shall fail, indicating WIPG0204.
- 1555           • If a filter query is specified,
- 1556           – the query language specified in the *FilterQueryLanguage* operation parameter shall be  
1557 valid. If this is not satisfied, the operation shall fail, indicating WIPG0221.
- 1558           – the query specified in the *FilterQueryString* operation parameter shall be a valid query in  
1559 the query language specified in the *FilterQueryLanguage* operation parameter. If this is not  
1560 satisfied, the operation shall fail, indicating WIPG0222 or WIPG0223.
- 1561           – the *AssociationClassName* operation input parameter shall be non-NULL. If this is not  
1562 satisfied, the operation shall fail, indicating WIPG0208.

- 1563
- 1564
- 1565
- The *IncludedProperties* operation parameter, if supported by the WBEM protocol, shall only be specified with a non-NULL value if the *AssociationClassName* operation input parameter is also non-NULL. If this is not satisfied, the operation shall fail, indicating WIPG0208.
- 1566
- 1567
- 1568
- 1569
- The namespace of any returned instance paths shall exist. If it does not exist, the operation may fail, indicating WIPG0204. Note that cross-namespace association traversals may return instance paths in a server or namespace that is different from the server or namespace of the source instance.
- 1570
- 1571
- The creation class of any returned instance paths shall exist in their namespace. If it does not exist, the operation may fail, indicating WIPG0214.

1572 **Postconditions:**

- 1573
- The enumeration session shall have been established and opened.
- 1574
- 1575
- A first set of instances with their instance paths shall have been returned as described in the Description paragraph for this operation.
- 1576
- Requirements on ACID properties:
    - Atomicity: Required (related to the creation of an enumeration context that is maintained by the WBEM server)
    - Update Consistency: N/A
    - Isolation: Required at the level of single instances, as defined in 5.8.
    - Durability: Required (related to creation of an enumeration context that is maintained by the WBEM server)
- 1577
- 1578
- 1579
- 1580
- 1581
- 1582

1583 **Error messages:**

1584

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	For input namespace
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0242	Invalid timeout	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0235	Continuation on error not supported	Mandatory	Infrastructure, class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0237	Filter queries not supported by WBEM server infrastructure	Optional	Infrastructure	
WIPG0244	Filter queries not supported by class implementation	Optional	Class implem.	
WIPG0221	Unknown query language	Mandatory	Infrastructure, class implem.	
WIPG0222	Query language feature not supported	Mandatory	Infrastructure, class implem.	
WIPG0223	Invalid query	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0204	Namespace not found	Optional	Infrastructure	For namespace of returned instance paths
WIPG0214	Class not found	Optional	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1585

1586 **6.4.6 OpenQueryInstances**

1587 **Purpose:**

1588 Establish and open an enumeration session for enumerating the instances of a query result in a  
 1589 given namespace and optionally return a first set of their instance representations.

1590 **Operation Input Parameters:**

1591

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace in which the query is executed (Context Parameter)
QueryString	QueryString	Mandatory	Query string of a query that defines the set of instances to be returned
QueryLanguage	QueryLanguage	Mandatory	Query language of the query specified in <i>QueryString</i>
ReturnQueryResultClass	boolean	Mandatory	Indicates whether a class definition of the query result should be returned in <i>QueryResultClass</i>
OperationTimeout	uint32	Mandatory	Operation timeout, as defined in 6.4.2.4

Generic Name	Generic Type	Requirement	Description
ContinueOnError	boolean	Conditional	Indicates whether the enumeration session should be continued in case of error, as defined in 6.4.2.5 Condition: WBEM protocol supports client side control of continuation on error for pulled instance enumeration operations.
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.2.6

1592

1593 **Operation Output Parameters:**

1594

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecification [ ]	Mandatory	Sequence of the returned first set of instance representations
QueryResultClass	ClassSpecification	Mandatory	Representation of a class definition for the query result
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.2.1
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.2.2

1595

1596 **Description:**

1597 The *OpenQueryInstances* operation establishes and opens an enumeration session for enumerating  
 1598 the instances representing the result of the query specified in *QueryString* in the namespace  
 1599 referenced by *NamespacePath*. That enumeration session allows retrieving representations of these  
 1600 result instances through successive *PullInstances* operations (see 6.4.9). Retrieval of a first set of  
 1601 those instances may be requested by setting *MaxObjectCount* to a value > 0.

1602 The set of instances to be returned (as instance representations) is the first set of instances from the  
 1603 set of instances to be enumerated throughout the entire enumeration session, such that no more  
 1604 than *MaxObjectCount* instances are returned. Returning no instances in the *InstanceList* operation  
 1605 parameter does not imply that the enumeration session has been exhausted. Only the  
 1606 *EndOfSequence* operation output parameter indicates whether the enumeration session has been  
 1607 exhausted.

1608 The returned instance representations have no corresponding addressable instances that exist.

1609 If *QueryLanguage* is not NULL, it shall specify a valid query language and *QueryString* shall be a  
 1610 valid query in that query language. Neither the query language nor the format of the filter query is  
 1611 defined by this specification. Conformant WBEM protocols shall specify a mechanism for determining  
 1612 the set of query languages that are valid for *QueryLanguage*. The simplest way to do this is to list the  
 1613 set of valid query languages.

1614 The value of the *ReturnQueryResultClass* operation input parameter controls whether or not a class  
 1615 definition is returned in the *QueryResultClass* operation output parameter. If FALSE, then  
 1616 *QueryResultClass* shall be NULL. If TRUE, then the value of *QueryResultClass* shall be a class  
 1617 definition that defines the properties of each instance of the query result. The name of this class shall

1618 be CIM\_QueryResult. This class is only a representation of a class that has no corresponding  
 1619 addressable class residing in the WBEM server.

1620 **Preconditions:**

- 1621 • The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation  
 1622 shall fail, indicating WIPG0204.
- 1623 • The query language specified in the *QueryLanguage* operation parameter shall be a valid query  
 1624 language. If this is not satisfied, the operation shall fail, indicating WIPG0221.
- 1625 • The query specified in the *QueryString* operation parameter shall be a valid query in the query  
 1626 language specified in the *QueryLanguage* operation parameter. If this is not satisfied, the  
 1627 operation shall fail, indicating WIPG0222 or WIPG0223.

1628 **Postconditions:**

- 1629 • The enumeration session shall have been established and opened.
- 1630 • A first set of instances shall have been returned as described in the Description paragraph for  
 1631 this operation.
- 1632 • Requirements on ACID properties:
  - 1633 – Atomicity: Required (related to the creation of an enumeration context that is maintained by  
 1634 the WBEM server)
  - 1635 – Update Consistency: N/A
  - 1636 – Isolation: Required at the level of single instances, as defined in 5.8.
  - 1637 – Durability: Required (related to creation of an enumeration context that is maintained by  
 1638 the WBEM server)

1639 **Error messages:**

1640

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0242	Invalid timeout	Mandatory	Infrastructure, class implem.	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	



Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0235	Continuation on error not supported	Mandatory	Infrastructure, class implem.	
WIPG0221	Unknown query language	Mandatory	Infrastructure, class implem.	
WIPG0222	Query language feature not supported	Mandatory	Infrastructure, class implem.	
WIPG0223	Invalid query	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1641

## 1642 6.4.7 Common operation parameters for the pull operations

1643 This subclause defines commonly used operation parameters for the Pull operations. The description of  
 1644 the individual Pull operations references these operation parameters as appropriate. However, not every  
 1645 Pull operation uses every one of these common operation parameters.

### 1646 6.4.7.1 NamespacePath

1647 The *NamespacePath* operation input parameter references the namespace identified by the context  
 1648 parameter of the Open operation that established and opened the enumeration session.

### 1649 6.4.7.2 EnumerationContext

1650 The *EnumerationContext* operation input/output parameter is the enumeration context value representing  
 1651 the enumeration session to be used.

1652 Support for the *EnumerationContext* operation parameter in a conformant WBEM protocol is mandatory.

1653 When invoking the Pull operation, the enumeration session represented by *EnumerationContext* shall be  
 1654 open. The enumeration session shall have been established using one of the Open operations whose  
 1655 type of enumerated element matches the Pull operation. For the first Pull operation on an enumeration  
 1656 session, the value of *EnumerationContext* shall be the enumeration context value returned by a  
 1657 successful Open operation that established and opened that enumeration session. For any subsequent  
 1658 Pull operations on that enumeration session, the value of *EnumerationContext* shall be the value of  
 1659 *EnumerationContext* as returned by the previous Pull operation on the same enumeration session.

1660 After completing the Pull operation, the enumeration session represented by *EnumerationContext* shall  
 1661 be open or closed.

### 1662 6.4.7.3 EndOfSequence

1663 The *EndOfSequence* operation output parameter when used in Pull operations behaves as defined in  
 1664 6.4.2.2

1665 **6.4.7.4 MaxObjectCount**

1666 The *MaxObjectCount* operation input parameter when used in Pull operations behaves as defined in  
 1667 6.4.2.6.

1668 **6.4.8 PullInstancesWithPath**

1669 **Purpose:**

1670 Retrieve the next set of instance representations and instance paths from an open enumeration  
 1671 session.

1672 **Operation Input Parameters:**

1673

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace for the enumeration, as defined in 6.4.7.1 (Context Parameter)
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.7.2
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.7.4

1674

1675 **Operation Output Parameters:**

1676

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecificationWithPath [ ]	Mandatory	Next set of returned instance representations and instance paths
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.7.2
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.7.3

1677

1678 **Description:**

1679 The *PullInstancesWithPath* operation retrieves the next set of instance representations and instance  
 1680 paths from an open enumeration session.

1681 The enumeration session shall have been established using one of the following operations:

- 1682 • OpenEnumerateInstances
- 1683 • OpenAssociators
- 1684 • OpenReferences

1685 The set of instances to be returned (as instance representations and instance paths) is the next set  
 1686 of instances from the set of instances to be enumerated throughout the entire enumeration session,  
 1687 such that no more than *MaxObjectCount* instances are returned. Returning no instances does not

1688 imply that the enumeration session has been exhausted. Only the *EndOfSequence* operation output  
1689 parameter indicates whether the enumeration session has been exhausted.

1690 The set of properties to be included in any retrieved instances shall be the as determined using the  
1691 Open operation that established the enumeration session.

#### 1692 **Preconditions:**

- 1693 • The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation  
1694 shall fail, indicating WIPG0204.
- 1695 • The enumeration session identified by *EnumerationContext* shall be open. If this is not satisfied,  
1696 the operation shall fail, indicating WIPG0241.
- 1697 • The value of *EnumerationContext* shall be the enumeration context value returned by the  
1698 previous Open or Pull operation on the same enumeration session. If this is not satisfied, the  
1699 operation shall fail, indicating WIPG0241.
- 1700 • The namespace of any returned instance paths shall exist. If it does not exist, the operation may  
1701 fail, indicating WIPG0204. Note that cross-namespace association traversals may return  
1702 instance paths in a server or namespace that is different from the server or namespace of the  
1703 source instance.
- 1704 • The creation class of any returned instance paths shall exist in their namespace. If it does not  
1705 exist, the operation may fail, indicating WIPG0214.

#### 1706 **Postconditions:**

- 1707 • The set of instances with their instance paths shall have been returned as described in the  
1708 Description paragraph for this operation.
- 1709 • Requirements on ACID properties:
  - 1710 – Atomicity: Required (related to updates to an enumeration context that is maintained by the  
1711 WBEM server)
  - 1712 – Update Consistency: N/A
  - 1713 – Isolation: Required at the level of single instances, as defined in 5.8.
  - 1714 – Durability: Required (related to updates to an enumeration context that is maintained by  
1715 the WBEM server)

#### 1716 **Error messages:**

1717

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0241	Invalid enumeration context	Mandatory	Class implem.	
WIPG0238	Pull operation has been abandoned due to enumeration context closure	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1718

1719 **6.4.9 PullInstances**

1720 **Purpose:**

1721 Retrieve the next set of instances from an open enumeration session.

1722 **Operation Input Parameters:**

1723

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace for the enumeration, as defined in 6.4.7.1 (Context Parameter)
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.7.2
MaxObjectCount	uint32	Mandatory	Maximum number of instances that may be returned by this operation, as defined in 6.4.7.4

1724

1725 **Operation Output Parameters:**

1726

Generic Name	Generic Type	Requirement	Description
InstanceList	InstanceSpecification []	Mandatory	Next set of returned instance representations
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.7.2
EndOfSequence	boolean	Mandatory	Indicates end of sequence for the enumeration session, as defined in 6.4.7.3

1727

1728 **Description:**

1729 The *PullInstances* operation retrieves the next set of instance representations without their instance  
1730 paths from an open enumeration session.

1731 The enumeration session shall have been established using one of the following operations:

- 1732 • *OpenQueryInstances*

1733 The set of instances to be returned (as instance representations) is the next set of instances from the  
1734 set of instances to be enumerated throughout the entire enumeration session, such that no more  
1735 than *MaxObjectCount* instances are returned. Returning no instances does not imply that the  
1736 enumeration session has been exhausted. Only the *EndOfSequence* operation output parameter  
1737 indicates whether the enumeration session has been exhausted.

1738 The set of properties to be included in any retrieved instances shall be the as determined using the  
1739 Open operation that established the enumeration session.

1740 **Preconditions:**

- 1741 • The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation  
1742 shall fail, indicating WIPG0204.
- 1743 • The enumeration session identified by *EnumerationContext* shall be open. If this is not satisfied,  
1744 the operation shall fail, indicating WIPG0241.
- 1745 • The value of *EnumerationContext* shall be the enumeration context value returned by the  
1746 previous Open or Pull operation on the same enumeration session. If this is not satisfied, the  
1747 operation shall fail, indicating WIPG0241.

1748 **Postconditions:**

- 1749 • The set of instances shall have been returned as described in the Description paragraph for this  
1750 operation.
- 1751 • Requirements on ACID properties:
  - 1752 – Atomicity: Required (related to updates to an enumeration context that is maintained by the  
1753 WBEM server)
  - 1754 – Update Consistency: N/A
  - 1755 – Isolation: Required at the level of single instances, as defined in 5.8.
  - 1756 – Durability: Required (related to updates to an enumeration context that is maintained by  
1757 the WBEM server)

1758 **Error messages:**

1759

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0241	Invalid enumeration context	Mandatory	Class implem.	
WIPG0238	Pull operation has been abandoned due to enumeration context closure	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1760

1761 **6.4.10 CloseEnumeration**

1762 **Purpose:**

1763 Close an open enumeration session.

1764 **Operation Input Parameters:**

1765

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace for the enumeration, as defined in 6.4.7.1 (Context Parameter)
EnumerationContext	EnumerationContext	Mandatory	Enumeration context value, as defined in 6.4.7.2

1766

1767 **Operation Output Parameters:**

1768 None.

1769 **Description:**

1770 The *CloseEnumeration* operation closes the open enumeration session identified by  
1771 *EnumerationContext*.

1772 The enumeration session shall have been established using any of the Open operations.

1773 Enumeration sessions are closed implicitly when exhausted, so this operation only needs to be used  
1774 when terminating an enumeration sequence before it is exhausted.

1775 **Preconditions:**

- 1776 • The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation  
1777 shall fail, indicating WIPG0204.
- 1778 • The enumeration session identified by *EnumerationContext* shall be open. If this is not satisfied,  
1779 the operation shall fail, indicating WIPG0241.
- 1780 • The value of *EnumerationContext* shall be the enumeration context value returned by the  
1781 previous Open or Pull operation on the same enumeration session. If this is not satisfied, the  
1782 operation shall fail, indicating WIPG0241.

1783 **Postconditions:**

- 1784 • The enumeration session identified by *EnumerationContext* is closed.
- 1785 • Requirements on ACID properties:
  - 1786 – Atomicity: Required (related to updates to or deletion of an enumeration context that is  
1787 maintained by the WBEM server)
  - 1788 – Update Consistency: N/A
  - 1789 – Isolation: Required
  - 1790 – Durability: Required (related to updates to or deletion of an enumeration context that is  
1791 maintained by the WBEM server)

1792 **Error messages:**

1793

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0228	Operation not supported by class implementation	Mandatory	Class implem.	
WIPG0241	Invalid enumeration context	Mandatory	Class implem.	
WIPG0239	Pull operation cannot be abandoned	Mandatory	Class implem.	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1794

1795 **6.5 Method invocation operations**

1796 This subclause defines server operations for the invocation of CIM methods.

1797 **6.5.1 InvokeMethod**

1798 **Purpose:**

1799 Invoke a non-static method on an instance.

1800 **Operation Input Parameters:**

1801

Generic Name	Generic Type	Requirement	Description
InstancePath	InstancePath	Mandatory	Instance path of the instance the method is invoked on (Context Parameter)
MethodName	MethodName	Mandatory	Name of the method being invoked
InParmValues	ParameterValue [ ]	Mandatory	Unordered set of named input parameter values of the method

1802

1803 **Operation Output Parameters:**

1804

Generic Name	Generic Type	Requirement	Description
OutParmValues	ParameterValue [ ]	Mandatory	Unordered set of named output parameter values of the method
ReturnValue	ReturnValue	Mandatory	Return value of the method

1805

1806 **Description:**

1807 Invoke a CIM method using an instance path. The method may be static or non-static.

1808 Conformant WBEM protocols shall define a mapping for the invocation of CIM methods using an  
 1809 instance path, including a mapping of the operation parameters defined in the tables above. These  
 1810 rules may map the method invocation to a single operation, map each method to its own separate  
 1811 operation, or define any other appropriate mapping.

1812 If the implementation of the method could be invoked, the operation is considered successful,  
 1813 regardless of what the semantics of any return values or output parameters is. For example, if a  
 1814 method defines that a particular return value indicates an error condition, the method invocation was  
 1815 still successful from a perspective of the invocation operation.



1816 **Preconditions:**

- 1817       • The instance referenced by *InstancePath* shall exist. If it does not exist, the operation shall fail,  
1818       indicating WIPG0213.
- 1819       • The creation class of the instance referenced by *InstancePath* shall exist. If it does not exist, the  
1820       operation shall fail, indicating WIPG0214.
- 1821       • The namespace of the instance referenced by *InstancePath* shall exist. If it does not exist, the  
1822       operation shall fail, indicating WIPG0204.
- 1823       • The method to be invoked shall be exposed by the creation class of the instance referenced by  
1824       *InstancePath*. If this is not satisfied, the operation shall fail, indicating WIPG0218.

1825 **Postconditions:**

- 1826       • The CIM method shall have been invoked.
- 1827       • Requirements on ACID properties:
- 1828       – Atomicity: Recommended
- 1829       – Update Consistency: Recommended
- 1830       – Isolation: Recommended
- 1831       – Durability: Required

1832 **Error messages:**

1833

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0229	Method invocation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0218	No such method	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0213	Instance not found	Mandatory	Class implem.	
WIPG0219	Method not supported by class implementation	Mandatory	Class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1834

1835 **6.5.2 InvokeStaticMethod**

1836 **Purpose:**

1837 Invoke a static method on a class.

1838 **Operation Input Parameters:**

1839

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the class the method is invoked on (Context Parameter)
MethodName	MethodName	Mandatory	Name of the method being invoked
InParmValues	ParameterValue [ ]	Mandatory	Unordered set of named input parameter values of the method

1840

1841 **Operation Output Parameters:**

1842

Generic Name	Generic Type	Requirement	Description
OutParmValues	ParameterValue [ ]	Mandatory	Unordered set of named output parameter values of the method
ReturnValue	ReturnValue	Mandatory	Return value of the method

1843

1844 **Description:**

1845 Invoke a static CIM method using a class path.

1846 Conformant WBEM protocols shall define a mapping for the invocation of CIM methods using a class  
 1847 path, including a mapping of the operation parameters defined in the tables above. These rules may  
 1848 map the method invocation to a single operation, map each method to its own separate operation, or  
 1849 define any other appropriate mapping.

1850 If the implementation of the method could be invoked, the operation is considered successful,  
 1851 regardless of what the semantics of any return values or output parameters is. For example, if a  
 1852 method defines that a particular return value indicates an error condition, the method invocation was  
 1853 still successful from a perspective of the invocation operation.

1854 **Preconditions:**

- 1855 • The class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail,  
1856 indicating WIPG0214.
- 1857 • The namespace of the class referenced by *ClassPath* shall exist. If it does not exist, the  
1858 operation shall fail, indicating WIPG0204.
- 1859 • The method to be invoked shall be exposed by the creation class of the instance referenced by  
1860 *InstancePath*. If this is not satisfied, the operation shall fail, indicating WIPG0218.

1861 **Postconditions:**

- 1862 • The CIM method shall have been invoked.
- 1863 • Requirements on ACID properties:
  - 1864 – Atomicity: Recommended
  - 1865 – Update Consistency: Recommended
  - 1866 – Isolation: Recommended
  - 1867 – Durability: Required

1868 **Error messages:**

1869

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0229	Method invocation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0218	No such method	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Class implem.	
WIPG0219	Method not supported by class implementation	Mandatory	Class implem.	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1870

1871 **6.6 Class operations**

1872 This subclause defines server operations that target a single class or create a class. These operations  
 1873 include dealing with qualifier values defined on classes and their elements.

1874 **6.6.1 GetClass**

1875 **Purpose:**

1876 Retrieve a class.

1877 **Operation Input Parameters:**

1878

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the class to be retrieved (Context Parameter)
IncludeInheritedElements	boolean	Optional	Indicates whether any elements inherited from superclasses are to be included in the returned class
IncludeQualifiers	boolean	Mandatory	Indicates whether qualifier values on any returned CIM elements are to be included, as defined in 6.2.1

1879

1880 **Operation Output Parameters:**

1881

Generic Name	Generic Type	Requirement	Description
Class	ClassSpecification	Mandatory	Retrieved class representation

1882

1883 **Description:**

1884 The *GetClass* operation retrieves a representation of the class referenced by *ClassPath*.

1885 The set of properties to be included in the retrieved class shall be determined using the following  
 1886 algorithm:

- 1887 • Initially, the set of properties to be included is the set of properties exposed by the class to  
 1888 be retrieved. This includes all the duplicates of any duplicate non-overridden properties.
- 1889 • If *IncludeInheritedElements* is FALSE, it acts as a restricting filter on the elements  
 1890 (properties, methods, qualifiers) to be included in the returned class such that any  
 1891 elements inherited into the class to be retrieved are removed from the set of properties to  
 1892 be included. This is also known as reducing the elements to *local-only* elements.

1893 **Preconditions:**

- 1894
- The class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0214.
- 1895
- The namespace of the class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail, indicating WIPG0204.
- 1896
- 1897

1898 **Postconditions:**

- 1899
- The class representation shall have been returned as defined in the Description paragraph for this operation.
- 1900
- Requirements on ACID properties:
- 1901
- Atomicity: N/A
- 1902
- Update Consistency: N/A
- 1903
- Isolation: Required
- 1904
- Durability: N/A
- 1905

1906 **Error messages:**

1907

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

1908

1909 **6.6.2 DeleteClass**1910 **Purpose:**

1911 Delete a given class.

1912 **Operation Input Parameters:**

1913

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the class to be deleted (Context Parameter)
DeleteDependents	Boolean	Optional	<b>EXPERIMENTAL:</b> Indicates whether dependent classes and instances are to be deleted as well

1914

1915 **Operation Output Parameters:**

1916 None.

1917 **Description:**1918 The *DeleteClass* operation deletes the class referenced by *ClassPath*.

1919

---

 1920 **EXPERIMENTAL**

- 1921 If the WBEM protocol supports the *DeleteDependents* operation parameter, the following rules apply:
- 1922 • If *DeleteDependents* is TRUE, any classes that depend on the class referenced by
  - 1923 *ClassPath* in the way described below shall be deleted, and any instances of the class
  - 1924 referenced by *ClassPath* and of any classes depending on it shall be deleted according to
  - 1925 the rules defined for the
  - 1926 • DeleteInstance operation. If these rules cause the rejection of an instance deletion, the
  - 1927 • DeleteClass operation shall fail.
  - 1928 • If *DeleteDependents* is FALSE, the *DeleteClass* operation shall fail if any classes exist that
  - 1929 depend on the class referenced by *ClassPath* in the way described below, or if the class
  - 1930 referenced by *ClassPath* has any instances.

---

 1931 **EXPERIMENTAL**

- 1932 If the WBEM protocol does not support the *DeleteDependents* operation parameter, the *DeleteClass*
- 1933 operation shall fail if any classes exist that depend on the class referenced by *ClassPath* in the way
- 1934 described below, or if the class referenced by *ClassPath* has any instances.
- 1935 For the purpose of the *DeleteClass* operation, the following classes are considered depending on the
- 1936 class referenced by *ClassPath*:
- 1937 • Any subclasses of any class depending on the class referenced by *ClassPath*.
  - 1938 • Any association classes referencing any class depending on the class referenced by
  - 1939 *ClassPath*.
  - 1940 • Any classes defining a method with a parameter or a return value that is
  - 1941 – a reference to any class depending on the class referenced by *ClassPath*, or
  - 1942 – an embedded instance of any class depending on the class referenced by *ClassPath*,
  - 1943 or
  - 1944 – an embedded class depending on the class referenced by *ClassPath*.
  - 1945 • Any classes defining a property that is
  - 1946 – an embedded instance of any class depending on the class referenced by *ClassPath*,
  - 1947 or
  - 1948 – an embedded class depending on the class referenced by *ClassPath*.
- 1949 Any classes or instances that are automatically deleted may reside in a different namespace (which
- 1950 may reside in a different WBEM server) than the class referenced by *ClassPath*.
- 1951 In case of error, the consistency requirements defined in [DSP0004](#) cannot be guaranteed, but should
- 1952 be attempted to be satisfied in a best effort approach. In case of error, only a subset of the elements
- 1953 to be deleted may have been deleted, but each element shall have either been deleted completely or
- 1954 not at all. Also, classes shall only be deleted if all of its instances could be deleted successfully.
- 1955 NOTE In a non-transactional implementation, this requires an order of deletion that starts with those
- 1956 elements that do not depend on the deletion of other elements.

1957 **Preconditions:**

- 1958 • The class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail,  
1959 indicating WIPG0214.
- 1960 • The namespace of the class referenced by *ClassPath* shall exist. If it does not exist, the  
1961 operation shall fail, indicating WIPG0204.

1962 **Postconditions:**

- 1963 • The class referenced by *ClassPath* shall have been deleted.
- 1964 • If *DeleteDependents* was TRUE:
  - 1965 – any dependent classes and instances shall have been deleted as defined in the  
1966 Description paragraph for this operation, and
  - 1967 – any management profile defined implicit deletions of other instances shall have  
1968 happened, and
  - 1969 – any management profile defined effects of the deletion of all of these instances on any  
1970 managed resources shall have happened.
- 1971 • The consistency requirements defined in [DSP0004](#) shall be satisfied for any classes and  
1972 instances related to the deleted classes and instances.
- 1973 • Requirements on ACID properties:
  - 1974 – Atomicity: Required, if dependent classes and instances are handled by rejection, as  
1975 defined in 5.8.9. Recommended, if dependent classes and instances are handled by  
1976 delete propagation, as defined in 5.8.9.
  - 1977 – Update Consistency: Required, if dependent classes and instances are handled by  
1978 rejection, as defined in 5.8.9. Recommended, if dependent classes and instances are  
1979 handled by delete propagation, as defined in 5.8.9.
  - 1980 – Isolation: Required, if dependent classes and instances are handled by rejection, as  
1981 defined in 5.8.9. Recommended, if dependent classes and instances are handled by  
1982 delete propagation, as defined in 5.8.9.
  - 1983 – Durability: Required

1984 **Error messages:**

1985

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure, class implem.	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	



Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure, class implem.	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0224	Class has subclasses	Mandatory	Infrastructure	
WIPG0225	Class has instances	Mandatory	Infrastructure, class implem.	
WIPG0230	Class has referencing association classes	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure, class implem.	
WIPG0227	Other failure	Optional	Infrastructure, class implem.	

1986

1987 **6.6.3 ModifyClass**1988 **Purpose:**

1989 Change a given class.

1990 **Operation Input Parameters:**

1991

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the class to be changed. (Context Parameter)
ModifiedClass	ClassSpecification	Mandatory	Class representation specifying the new class definition

1992

1993 **Operation Output Parameters:**

1994 None.

1995 **Description:**1996 The *ModifyClass* operation changes the definition of the class referenced by *ClassPath*.

- 1997  
1998 Within the restrictions specified in the preconditions, the definition of the class referenced by *ClassPath* is replaced with the definition specified in *ModifiedClass*, as follows:
- 1999 • Any elements previously defined in the class to be changed (including overriding elements)  
2000 that are not specified in *ModifiedClass* shall be removed from the class to be changed.
  - 2001 • Any elements previously defined in the class to be changed (including overriding elements)  
2002 that are also specified in *ModifiedClass* shall be replaced with the definition from  
2003 *ModifiedClass*.
  - 2004 • Any elements not previously defined in the class to be changed (including overriding  
2005 elements) that are specified in *ModifiedClass* shall be added to the class to be changed, as  
2006 defined in *ModifiedClass*.
- 2007 Any instances whose creation class is the class referenced by *ClassPath* or one of its subclasses  
2008 shall be changed to reflect the changes to the class, as follows:
- 2009 • Added properties are reflected using the rules defined in the *ModifyInstance* operation  
2010 when processing a list of these new properties with their values set to their class defined  
2011 default values, or NULL where no class defined default value is defined.
- 2012 Any other changes to the class that are compatible with the preconditions do not affect existing  
2013 instances, for the following reasons:
- 2014 • A compatible removal of properties from a class can only happen for overridden properties  
2015 or for properties that move to a superclass, both of which is equivalent to potential changes  
2016 of qualifier values and the default property value. Changes of qualifier values do not affect  
2017 instances. A changed default value only affects new instances, but not existing instances.
  - 2018 • A compatible change of existing property definitions can only include potential changes of  
2019 qualifier values and the default property value. Changes of qualifier values do not affect  
2020 instances. A changed default value only affects new instances, but not existing instances.
  - 2021 • A compatible change of values of class qualifiers does not affect instances of the class.
  - 2022 • A compatible change to a method definition does not affect instances of the class.
- 2023 **Preconditions:**
- 2024 • The class referenced by *ClassPath* shall exist. If it does not exist, the operation shall fail,  
2025 indicating WIPG0214.
  - 2026 • The namespace of the class referenced by *ClassPath* shall exist. If it does not exist, the  
2027 operation shall fail, indicating WIPG0204.
  - 2028 • The name of the class defined by *ModifiedClass* shall be the name of the class referenced by  
2029 *ClassPath*. If this is not satisfied, the operation shall fail, indicating WIPG0208.
  - 2030 • If the class referenced by *ClassPath* has a superclass, the class defined by *ModifiedClass* shall  
2031 specify a superclass with the same name as that superclass. If the class referenced by  
2032 *ClassPath* has no superclass, the class defined by *ModifiedClass* shall not specify a superclass.  
2033 If this is not satisfied, the operation shall fail, indicating WIPG0226.
  - 2034 • The class defined by *ModifiedClass* shall only specify elements that when applied to the class to  
2035 be modified, result in a class definition that satisfies any consistency and backward compatibility  
2036 requirements defined in [DSP0004](#). For example, qualifiers with flavor *DisableOverride* shall not  
2037 be overridden, or data types of overridden properties shall not be changed. If this is not  
2038 satisfied, the operation shall fail, indicating WIPG0231.

2039 **Postconditions:**

- 2040
- 2041
- The definition of the class referenced by *ClassPath* shall have been modified as defined in the Description paragraph for this operation.
- 2042
- 2043
- Any instances of the class or its subclasses shall have been changed as defined in the Description paragraph for this operation.
- 2044
- 2045
- The consistency and backward compatibility requirements defined in [DSP0004](#) shall be satisfied for the modified class.
- 2046
- Requirements on ACID properties:
    - Atomicity: Required
    - Update Consistency: Required
    - Isolation: Required
    - Durability: Required
- 2047
- 2048
- 2049
- 2050

2051 **Error messages:**

2052

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0226	Superclass not found	Mandatory	Infrastructure	
WIPG0231	Incompatible class modification	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2053

2054 **6.6.4 CreateClass**2055 **Purpose:**

2056 Create a class.

2057 **Operation Input Parameters:**

2058

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace the class is to be created in (Context Parameter)
NewClass	ClassSpecification	Mandatory	Representation of the class to be created

2059

2060 **Operation Output Parameters:**

2061

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the new class

2062

2063 **Description:**

2064 The *CreateClass* operation creates a class in the namespace referenced by *NamespacePath*, using  
 2065 the class representation in *NewClass*, and returns the class path of the new class.

2066 If properties or methods defined in *NewClass* are intended to override properties or methods defined  
 2067 in a superclass of *NewClass*, then they shall define an *OVERRIDE* qualifier in their definition in  
 2068 *NewClass*. The *CreateClass* operation shall not add such qualifiers automatically.

2069 **Preconditions:**

- 2070 • The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation  
 2071 shall fail, indicating WIPG0204.
- 2072 • The class to be created shall not exist in the namespace referenced by *NamespacePath*. If this  
 2073 is not satisfied, the operation shall fail, indicating WIPG0217.
- 2074 • If *NewClass* specifies a superclass, that superclass shall exist in the namespace referenced by  
 2075 *NamespacePath*. If this is not satisfied, the operation shall fail, indicating WIPG0226.
- 2076 NOTE [DSP0004](#) does not provide for inheritance relationships that cross namespace boundaries.
- 2077 • The definition of *NewClass* shall satisfy any consistency requirements defined in [DSP0004](#). If  
 2078 this is not satisfied, the operation shall fail, indicating WIPG0208.

2079 **Postconditions:**

- 2080 • The class shall have been created as defined in the Description paragraph for this operation.
- 2081 • Requirements on ACID properties:
  - 2082 – Atomicity: Required
  - 2083 – Update Consistency: Required
  - 2084 – Isolation: Required
  - 2085 – Durability: Required

2086  
2087**Error messages:**

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0217	Class already exists	Mandatory	Infrastructure	
WIPG0226	Superclass not found	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2088

2089 **6.7 Class enumeration operations**2090 This subclause defines server operations that enumerate classes and return their representations and  
2091 class paths.2092 **6.7.1 EnumerateClasses**2093 **Purpose:**

2094 Enumerate classes in a namespace and return these classes together with their class paths.

2095 **Operation Input Parameters:**

2096

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace the enumeration is executed on (Context Parameter)
ClassName	ClassName	Mandatory	Optional: Name of the CIM class whose subclasses are to be enumerated. If not specified, top classes are enumerated.
IncludeSubclasses	boolean	Mandatory	Indicates whether the entire tree of subclasses is to be included in the result set, in addition

Generic Name	Generic Type	Requirement	Description
IncludeInheritedElements	boolean	Mandatory	Indicates whether any elements inherited from superclasses of <i>ClassName</i> are to be included in the returned classes
IncludeQualifiers	boolean	Mandatory	Indicates whether qualifier values on any returned CIM elements are to be included, as defined in 6.2.1

2097

2098 **Operation Output Parameters:**

2099

Generic Name	Generic Type	Requirement	Description
ClassList	ClassSpecificationWithPath []	Mandatory	Sequence of the enumerated classes with their class paths

2100

2101 **Description:**

2102 The *EnumerateClasses* operation enumerates classes (including association and indication classes)  
2103 in the namespace specified in *NamespacePath* and returns their representations and class paths.

2104 *ClassName* and *IncludeSubclasses* together determine the set of classes in the result set. The set of  
2105 classes in the result set is determined using the following algorithm:

- 2106 1) *ClassName* is optional to be specified by the WBEM client (Note that *ClassName* is  
2107 mandatory to be supported by the WBEM protocol). If *ClassName* is not specified, the  
2108 result set initially contains all top classes (that is, classes that do not have a superclass) in  
2109 the namespace. If *ClassName* is specified, the result set initially contains the subclasses of  
2110 the class specified in *ClassName* (not including the class specified in *ClassName*).
- 2111 2) If *IncludeSubclasses* is TRUE, then all direct and indirect subclasses of the classes that  
2112 are so far in the result set are added to the result set. Otherwise, the result set is not  
2113 changed.

2114 If *IncludeInheritedElements* is TRUE, then the set of CIM elements in each returned class shall  
2115 consist of all elements exposed by that class. Otherwise, the set of CIM elements in each returned  
2116 class shall consist only of all elements defined in the class specified in *ClassName* (including  
2117 overriding elements).

2118 The consistency model defined in 5.8 applies.

2119 **Preconditions:**

- 2120 • The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation  
2121 shall fail, indicating WIPG0204.
- 2122 • If *ClassName* is specified, the specified CIM class shall exist in the namespace referenced by  
2123 *NamespacePath*. If this is not satisfied, the operation shall fail, indicating WIPG0214.

2124 **Postconditions:**

- 2125       • The enumerated classes with their class paths shall have been returned as defined in the  
2126       Description paragraph for this operation.
- 2127       • Requirements on ACID properties:
- 2128       – Atomicity: N/A
- 2129       – Update Consistency: N/A
- 2130       – Isolation: Required at the level of single classes, as defined in 5.8.
- 2131       – Durability: N/A

2132 **Error messages:**

2133

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0214	Class not found	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2134

2135 **6.7.2 AssociatorClasses**2136 **Purpose:**

2137       Enumerate the classes that are associated with a given source class and return their class  
2138       representations and class paths.

2139 **Operation Input Parameters:**

2140

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the source class (Context Parameter)
AssociationClassName	ClassName	Mandatory	NULL, or name of the association class, acting as a restricting filter on the associated classes
AssociatedClassName	ClassName	Mandatory	NULL, or name of the associated class on any far end of the association, acting as a restricting filter on the associated classes
SourceRoleName	PropertyName	Mandatory	NULL, or name of the role on the starting end of the association, acting as a restricting filter on the associated classes
AssociatedRoleName	PropertyName	Mandatory	NULL, or name of the role on any far end of the association, acting as a restricting filter on the associated classes
IncludeQualifiers	boolean	Mandatory	Indicates whether qualifier values on any returned CIM elements are to be included, as defined in 6.2.1

2141

2142 **Operation Output Parameters:**

2143

Generic Name	Generic Type	Requirement	Description
ClassList	ClassSpecificationWithPath [ ]	Mandatory	Sequence of the returned class representations and class paths

2144

2145 **Description:**

2146 The *AssociatorClasses* operation traverses an association from a given source class on a starting  
2147 end to classes on all of its far ends and returns the associated classes together with their class  
2148 paths.

2149 The set of associated classes to be enumerated shall be determined using the following algorithm:

- 2150 • Initially, the set of classes to be enumerated is the set of all classes associated to any of  
2151 the far ends of all associations referencing the starting class.
- 2152 • If the *AssociationClassName* operation input parameter is not NULL, it acts as a restricting  
2153 filter on the classes to be enumerated such that each class that is associated with the  
2154 starting class using an association class where the class or one of its superclasses does  
2155 not have the name specified in *AssociationClassName*, is removed from the set of classes  
2156 to be enumerated. There shall be no validity checking performed for the  
2157 *AssociationClassName* operation input parameter; if the specified class does not exist, the  
2158 operation shall succeed with an empty result (because the filter did not match).
- 2159 • If the *AssociatedClassName* operation input parameter is not NULL, it acts as a restricting  
2160 filter on the classes to be enumerated such that each class where the class or one of its  
2161 superclasses does not have the name specified in *AssociatedClassName*, is removed from  
2162 the set of classes to be enumerated. There shall be no validity checking performed for the



- 2163 *AssociatedClassName* operation input parameter; if the specified class does not exist, the  
2164 operation shall succeed with an empty result (because the filter did not match).
- 2165 NOTE Specifying a non-NULL value for *AssociatedClassName* ensures that the returned classes  
2166 have the class specified in *AssociatedClassName* as a common superclass.
- 2167 • If the *SourceRoleName* operation input parameter is not NULL, it acts as a restricting filter  
2168 on the classes to be enumerated such that each class that is associated with the starting  
2169 class using an association class that has a role name on its starting end that is not the role  
2170 name specified in *SourceRoleName*, is removed from the set of classes to be enumerated.  
2171 There shall be no validity checking performed for the *SourceRoleName* operation input  
2172 parameter; if the specified role does not exist, the operation shall succeed with an empty  
2173 result (because the filter did not match).
  - 2174 • If the *AssociatedRoleName* operation input parameter is not NULL, it acts as a restricting  
2175 filter on the classes to be enumerated such that each class that is associated with the  
2176 starting class using an association class that has a role name on the far end referencing  
2177 that class that is not the role name specified in *AssociatedRoleName*, is removed from the  
2178 set of classes to be enumerated. There shall be no validity checking performed for the  
2179 *AssociatedRoleName* operation input parameter; if the specified role does not exist, the  
2180 operation shall succeed with an empty result (because the filter did not match).
- 2181 The consistency model defined in 5.8 applies.
- 2182 The set of properties to be included in each returned associated class shall be determined using the  
2183 following algorithm:
- 2184 • The set of properties to be included is the set of properties exposed by the class. This  
2185 includes all the duplicates of any duplicate non-overridden properties.
- 2186 **Preconditions:**
- 2187 • The namespace of the source class referenced by *ClassPath* shall exist. If it does not exist, the  
2188 operation shall fail, indicating WIPG0204.
  - 2189 • The namespace of any returned classes shall exist. If it does not exist, the operation shall fail,  
2190 indicating WIPG0204. Note that cross-namespace association traversals may return classes in  
2191 a server or namespace that is different from the server or namespace of the source class.
- 2192 **Postconditions:**
- 2193 • The associated classes with their class paths shall have been returned as described in the  
2194 Description paragraph for this operation.
  - 2195 • Requirements on ACID properties:
    - 2196 – Atomicity: N/A
    - 2197 – Update Consistency: N/A
    - 2198 – Isolation: Required at the level of single classes, as defined in 5.8.
    - 2199 – Durability: N/A

2200 **Error messages:**

2201

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	For input namespace and namespace of returned class paths
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2202

2203 **6.7.3 ReferenceClasses**

2204 **Purpose:**

2205 Enumerate the association classes that reference a given source class and return their  
 2206 representations and class paths.

2207 **Operation Input Parameters:**

2208

Generic Name	Generic Type	Requirement	Description
ClassPath	ClassPath	Mandatory	Class path of the source class (Context Parameter)
AssociationClassName	ClassName	Mandatory	NULL, or name of the association class, acting as a restricting filter on the association classes
SourceRoleName	PropertyName	Mandatory	NULL, or name of the role on the starting end of the association, acting as a restricting filter on the association classes
IncludeQualifiers	boolean	Mandatory	Indicates whether qualifier values on any returned CIM elements are to be included, as defined in 6.2.1

2209

2210 **Operation Output Parameters:**

2211

Generic Name	Generic Type	Requirement	Description
ClassList	ClassSpecificationWithPath []	Mandatory	Sequence of the CIM association classes

2212

2213 **Description:**

2214 The *ReferenceClasses* operation traverses an association from a class on a starting end to classes  
 2215 on all of its far ends and returns the CIM association classes traversed together with their class  
 2216 paths.

2217 The set of association classes to be enumerated shall be determined using the following algorithm:

2218 • Initially, the set of classes to be enumerated is the set of all association classes referencing  
 2219 the starting class.

2220 • If the *AssociationClassName* operation input parameter is not NULL, it acts as a restricting  
 2221 filter on the classes to be enumerated such that each association class where the class or  
 2222 one of its superclasses does not have the name specified in *AssociationClassName*, is  
 2223 removed from the set of classes to be enumerated. There shall be no validity checking  
 2224 performed for the *AssociationClassName* operation input parameter; if the specified class  
 2225 does not exist, the operation shall succeed with an empty result (because the filter did not  
 2226 match).

2227 NOTE Specifying a non-NULL value for *AssociationClassName* ensures that the returned classes  
 2228 have the class specified in *AssociationClassName* as a common superclass.

2229 • If the *SourceRoleName* operation input parameter is not NULL, it acts as a restricting filter  
 2230 on the classes to be enumerated such that each association class that has a role name on  
 2231 its starting end that is not the role name specified in *SourceRoleName*, is removed from  
 2232 the set of classes to be enumerated. There shall be no validity checking performed for the  
 2233 *SourceRoleName* operation input parameter; if the specified role does not exist, the  
 2234 operation shall succeed with an empty result (because the filter did not match).

2235 The consistency model defined in 5.8 applies.

2236 The set of properties to be included in each returned association class shall be determined using the  
 2237 following algorithm:

2238 • The set of properties to be included is the set of properties exposed by the association  
 2239 class. This includes all the duplicates of any duplicate non-overridden properties.

2240 **Preconditions:**

2241 • The namespace of the source class referenced by *ClassPath* shall exist. If it does not exist, the  
 2242 operation shall fail, indicating WIPG0204.

2243 • The namespace of any returned classes shall exist. If it does not exist, the operation shall fail,  
 2244 indicating WIPG0204. Note that cross-namespace association traversals may return classes in  
 2245 a server or namespace that is different from the server or namespace of the source class.

2246 **Postconditions:**

- 2247 • The association classes with their class paths shall have been returned as described in the
- 2248 Description paragraph for this operation.
- 2249 • Requirements on ACID properties:
- 2250 – Atomicity: N/A
- 2251 – Update Consistency: N/A
- 2252 – Isolation: Required at the level of single classes, as defined in 5.8.
- 2253 – Durability: N/A

2254 **Error messages:**

2255

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	For input namespace and namespace of returned class paths
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2256

2257 **6.8 Qualifier type operations**

2258 This subclause defines server operations that deal with qualifier types. As defined in [DSP0004](#), qualifier

2259 types represent the declarations of qualifiers, not their values.

2260 **6.8.1 GetQualifierType**

2261 **Purpose:**

2262 Retrieve a qualifier type.

2263 **Operation Input Parameters:**

2264

Generic Name	Generic Type	Requirement	Description
QualifierTypePath	QualifierTypePath	Mandatory	Qualifier type path of the qualifier type to be retrieved (Context Parameter)

2265

2266 **Operation Output Parameters:**

2267

Generic Name	Generic Type	Requirement	Description
QualifierType	QualifierType	Mandatory	Representation of the returned qualifier type

2268

2269 **Description:**

2270 The *GetQualifierType* operation retrieves a representation of the qualifier type referenced by  
2271 *QualifierTypePath*.

2272 **Preconditions:**

- 2273 • The qualifier type referenced by *QualifierTypePath* shall exist. If it does not exist, the operation  
2274 shall fail, indicating WIPG0215.
- 2275 • The namespace of the qualifier type referenced by *QualifierTypePath* shall exist. If it does not  
2276 exist, the operation shall fail, indicating WIPG0204.

2277 **Postconditions:**

- 2278 • The representation of the qualifier type shall have been returned as described in the Description  
2279 paragraph for this operation.
- 2280 • Requirements on ACID properties:
  - 2281 – Atomicity: N/A
  - 2282 – Update Consistency: N/A
  - 2283 – Isolation: Required
  - 2284 – Durability: N/A

2285 **Error messages:**

2286

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0215	Qualifier type not found	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2287

2288 **6.8.2 DeleteQualifierType**

2289 **Purpose:**

2290 Delete a given qualifier type.

2291 **Operation Input Parameters:**

2292

Generic Name	Generic Type	Requirement	Description
QualifierTypePath	QualifierTypePath	Mandatory	Qualifier type path of the qualifier type to be deleted (Context Parameter)

2293

2294 **Operation Output Parameters:**

2295 None.

2296 **Description:**

2297 The *DeleteQualifierType* operation deletes the qualifier type referenced by *QualifierTypePath*.

2298 As defined in [DSP0004](#), any namespace needs to contain qualifier types for the meta qualifiers and  
 2299 standard qualifiers, and may contain qualifier types for the optional qualifiers. Thus, deleting any  
 2300 required qualifier types from a namespace will render that namespace non-compliant to [DSP0004](#).

2301 **Preconditions:**

- 2302 • The qualifier type referenced by *QualifierTypePath* shall exist. If it does not exist, the operation  
 2303 shall fail, indicating WIPG0215.
- 2304 • The namespace of the qualifier type referenced by *QualifierTypePath* shall exist. If it does not  
 2305 exist, the operation shall fail, indicating WIPG0204.
- 2306 • The qualifier identified by *QualifierTypePath* shall not be specified on any element in the same  
 2307 namespace. If this is not satisfied, the operation shall fail, indicating WIPG0233.

2308 **Postconditions:**

- 2309       • The qualifier type shall have been deleted as described in the Description paragraph for this  
2310       operation.
- 2311       • Requirements on ACID properties:
- 2312       – Atomicity: Required
- 2313       – Update Consistency: Required
- 2314       – Isolation: Required
- 2315       – Durability: Required

2316 **Error messages:**

2317

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0215	Qualifier type not found	Mandatory	Infrastructure	
WIPG0233	Qualifier type is used	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2318

2319 **6.8.3 ModifyQualifierType**2320 **Purpose:**

2321       Change a given qualifier type.

2322 **Operation Input Parameters:**

2323

Generic Name	Generic Type	Requirement	Description
QualifierTypePath	QualifierTypePath	Mandatory	Qualifier type path of the qualifier type to be changed (Context Parameter)
ModifiedQualifierType	QualifierType	Mandatory	Representation of the changed qualifier type

2324

2325 **Operation Output Parameters:**

2326 None.

2327 **Description:**

2328 The *ModifyQualifierType* operation changes the qualifier type referenced by *QualifierTypePath*.

2329 The qualifier type referenced by *QualifierTypePath* is replaced with the qualifier type representation  
2330 specified in *ModifiedQualifierType*.

2331 As defined in [DSP0004](#), any namespace needs to contain qualifier types for the meta qualifiers and  
2332 standard qualifiers, and may contain qualifier types for the optional qualifiers. Thus, changing these  
2333 qualifier types in a namespace inconsistently with their [DSP0004](#) definition will render that  
2334 namespace non-compliant to [DSP0004](#).

2335 **Preconditions:**

- 2336 • The qualifier type referenced by *QualifierTypePath* shall exist. If it does not exist, the operation  
2337 shall fail, indicating WIPG0215.
- 2338 • The namespace of the qualifier type referenced by *QualifierTypePath* shall exist. If it does not  
2339 exist, the operation shall fail, indicating WIPG0204.
- 2340 • The name of the qualifier type representation specified in *ModifiedQualifierType* shall equal the  
2341 name of the qualifier type referenced by *QualifierTypePath*. If this is not satisfied, the operation  
2342 shall fail, indicating WIPG0208.
- 2343 • The request to modify the qualifier type shall satisfy any backward compatibility requirements  
2344 defined in [DSP0004](#). If this is not satisfied, the operation shall fail, indicating WIPG0234.
- 2345 • If the qualifier type referenced by *QualifierTypePath* is one of the qualifiers defined in [DSP0004](#),  
2346 (i.e., meta, standard, and optional qualifiers), the new definition of the qualifier in  
2347 *ModifiedQualifierType* shall be consistent with the definition of the qualifier in [DSP0004](#). If this is  
2348 not satisfied, the operation shall fail, indicating WIPG0245.

2349 **Postconditions:**

- 2350 • The qualifier type referenced by *QualifierTypePath* shall have been modified as defined in the  
2351 Description paragraph for this operation.
- 2352 • The backward compatibility requirements defined in [DSP0004](#) shall be satisfied for the modified  
2353 qualifier type.
- 2354 • Requirements on ACID properties:
  - 2355 – Atomicity: Required
  - 2356 – Update Consistency: Required



- 2357           – Isolation: Required  
 2358           – Durability: Required

2359 **Error messages:**

2360

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0215	Qualifier type not found	Mandatory	Infrastructure	
WIPG0234	Incompatible modification of qualifier type	Mandatory	Infrastructure	
WIPG0245	Qualifier type inconsistent with <a href="#">DSP0004</a>	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2361

2362 **6.8.4 CreateQualifierType**

2363 **Purpose:**

2364           Create a qualifier type.

2365 **Operation Input Parameters:**

2366

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace in which the qualifier type is to be created (Context Parameter)
NewQualifierType	QualifierType	Mandatory	Representation of the qualifier type to be created

2367

2368 **Operation Output Parameters:**

2369

Generic Name	Generic Type	Requirement	Description
QualifierTypePath	QualifierTypePath	Mandatory	Qualifier type path of the new qualifier type

2370

2371 **Description:**

2372 The *CreateQualifierType* operation creates a qualifier type in the namespace referenced by  
 2373 *NamespacePath*, using the qualifier type representation specified in *NewQualifierType*, and returns  
 2374 the qualifier type path of the new qualifier type.

2375 As defined in [DSP0004](#), any namespace needs to contain qualifier types for the meta qualifiers and  
 2376 standard qualifiers, and may contain qualifier types for the optional qualifiers. Thus, creating these  
 2377 qualifier types in a namespace inconsistently with their [DSP0004](#) definition will render that  
 2378 namespace non-compliant to [DSP0004](#).

2379 **Preconditions:**

- 2380 • The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation  
 2381 shall fail, indicating WIPG0204.
- 2382 • The qualifier type to be created shall not exist in the namespace referenced by  
 2383 *NamespacePath*. If this is not satisfied, the operation shall fail, indicating WIPG0248.
- 2384 • If the qualifier type defined in *NewQualifierType* is one of the qualifiers defined in [DSP0004](#),  
 2385 (i.e., meta, standard, and optional qualifiers), the definition of the qualifier in *NewQualifierType*  
 2386 shall be consistent with the definition of the qualifier in [DSP0004](#). If this is not satisfied, the  
 2387 operation shall fail, indicating WIPG0245.

2388 **Postconditions:**

- 2389 • The qualifier type shall have been created as defined in the Description paragraph for this  
 2390 operation.
- 2391 • Requirements on ACID properties:
  - 2392 – Atomicity: Required
  - 2393 – Update Consistency: Required
  - 2394 – Isolation: Required
  - 2395 – Durability: Required

2396 **Error messages:**

2397

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0248	Qualifier type already exists	Mandatory	Infrastructure	
WIPG0245	Qualifier type inconsistent with <a href="#">DSP0004</a>	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2398

2399 **6.8.5 EnumerateQualifierTypes**2400 **Purpose:**

2401 Enumerate the qualifier types in a namespace.

2402 **Operation Input Parameters:**

2403

Generic Name	Generic Type	Requirement	Description
NamespacePath	NamespacePath	Mandatory	Namespace path of the namespace in which the qualifier types are to be enumerated (Context Parameter)

2404

2405 **Operation Output Parameters:**

2406

Generic Name	Generic Type	Requirement	Description
QualifierTypeList	QualifierTypeWithPath []	Mandatory	Sequence of the returned qualifier type representations and qualifier type paths

2407

2408 **Description:**

2409 The *EnumerateQualifierTypes* operation enumerates all qualifier types in the namespace referenced  
 2410 by *NamespacePath*, and returns their representations and qualifier type paths.

2411 **Preconditions:**

- 2412 • The namespace referenced by *NamespacePath* shall exist. If it does not exist, the operation  
 2413 shall fail, indicating WIPG0204.

2414 **Postconditions:**

- 2415 • The qualifier type representations and qualifier type paths shall have been returned as defined
- 2416 in the Description paragraph for this operation.
- 2417 • Requirements on ACID properties:
- 2418 – Atomicity: N/A
- 2419 – Update Consistency: N/A
- 2420 – Isolation: Required at the level of single qualifier types, as defined in 5.8.
- 2421 – Durability: N/A

2422 **Error messages:**

2423

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0236	WBEM server is shutting down	Optional	Infrastructure	
WIPG0240	WBEM server limits are exceeded	Optional	Infrastructure	
WIPG0204	Namespace not found	Mandatory	Infrastructure	
WIPG0203	Operation not supported by WBEM server infrastructure	Mandatory	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	
WIPG0227	Other failure	Optional	Infrastructure	

2424

2425 **6.9 Indication delivery operations**

2426 This subclause defines listener operations that deal with the delivery of indications.

2427 **6.9.1 DeliverIndication**

2428 **Purpose:**

2429 Deliver an indication to a listener.

2430 **Operation Input Parameters:**

2431

Generic Name	Generic Type	Requirement	Description
ListenerDestination	ListenerDestination	Mandatory	Address of the listener to which the indication will be delivered (see 5.4.20 for details) (Context Parameter)
Indication	InstanceSpecification	Mandatory	Representation of the indication instance

2432

2433 **Operation Output Parameters:**

2434 None

2435 **Description:**

2436 The *DeliverIndication* listener operation delivers the indication specified by *Indication* to the listener  
2437 referenced by *ListenerDestination*.

2438 Reliable indication delivery as defined in DSP1054 is an optional part of the operation semantics.  
2439 Generic operations mappings shall state whether reliable indication delivery is supported.

2440 **Preconditions:**

- 2441 • None

2442 **Postconditions:**

- 2443 • The indication shall have been delivered to the listener.
- 2444 • Requirements on ACID properties:
  - 2445 – Atomicity: N/A
  - 2446 – Update Consistency: N/A
  - 2447 – Isolation: N/A
  - 2448 – Durability: N/A

2449 **Error messages:**

2450

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0201	Access denied	Mandatory	Infrastructure	
WIPG0250	WBEM listener is shutting down	Optional	Infrastructure	
WIPG0251	WBEM listener limits are exceeded	Optional	Infrastructure	
WIPG0205	Missing input parameter	Mandatory	Infrastructure	
WIPG0206	Duplicate input parameter	Mandatory	Infrastructure	
WIPG0207	Unknown input parameter	Mandatory	Infrastructure	
WIPG0208	Incompatible input parameter type	Mandatory	Infrastructure	

Message ID	Message Name	Requirement	Sources	Additional Description
WIPG0249	Invalid input parameter value	Mandatory	Infrastructure	
WIPG0243	Timeout	Optional	Infrastructure	This also covers timeout due to exhaustion of retries in reliable indication delivery (if supported).
WIPG0227	Other failure	Optional	Infrastructure	

2451

## ANNEX A (normative)

### Cross-namespace associations

2456 This annex describes cross-namespace associations, in order to define which instances and classes exist  
 2457 in which namespace in such a scenario, and what is to be returned by association traversal operations.  
 2458 This annex reflects the preconditions stated for the association traversal operations in this specification,  
 2459 but it defines additional rules for conforming implementations and is therefore normative.

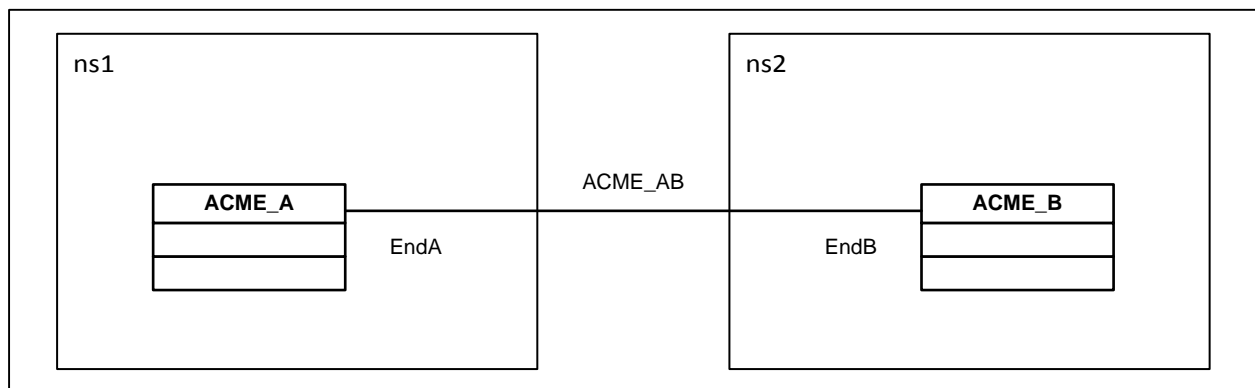
2460 In this annex, classes in a particular namespace are referred to using the syntax `<ns-name>::<class-`  
 2461 `name>` where `<ns-name>` is the namespace name and `<class-name>` is the class name. Instances in a  
 2462 particular namespace are referred to using the syntax `<ns-name>::<inst-name>` where `<ns-name>`  
 2463 is again the namespace name and `<inst-name>` is the name of the instance as stated in the diagram  
 2464 (which is purely a diagramming name and has nothing to do with its keys).

2465 In this version of this document, this annex only covers the simple case of a binary association where  
 2466 both sides use the same schema version. More complex cases, e.g. of associations with more than two  
 2467 ends, or with schema different versions, are possible, but not covered in this version.

#### 2468 A.1 Binary association using same schema version

2469 This subclause discusses a binary association (that is, an association with two ends) that crosses  
 2470 namespaces, and the two namespaces contain the same version of the schema.

2471 Figure 4 is a UML class diagram showing the classes used by this scenario, in the typical drawing  
 2472 notation used in management profiles:



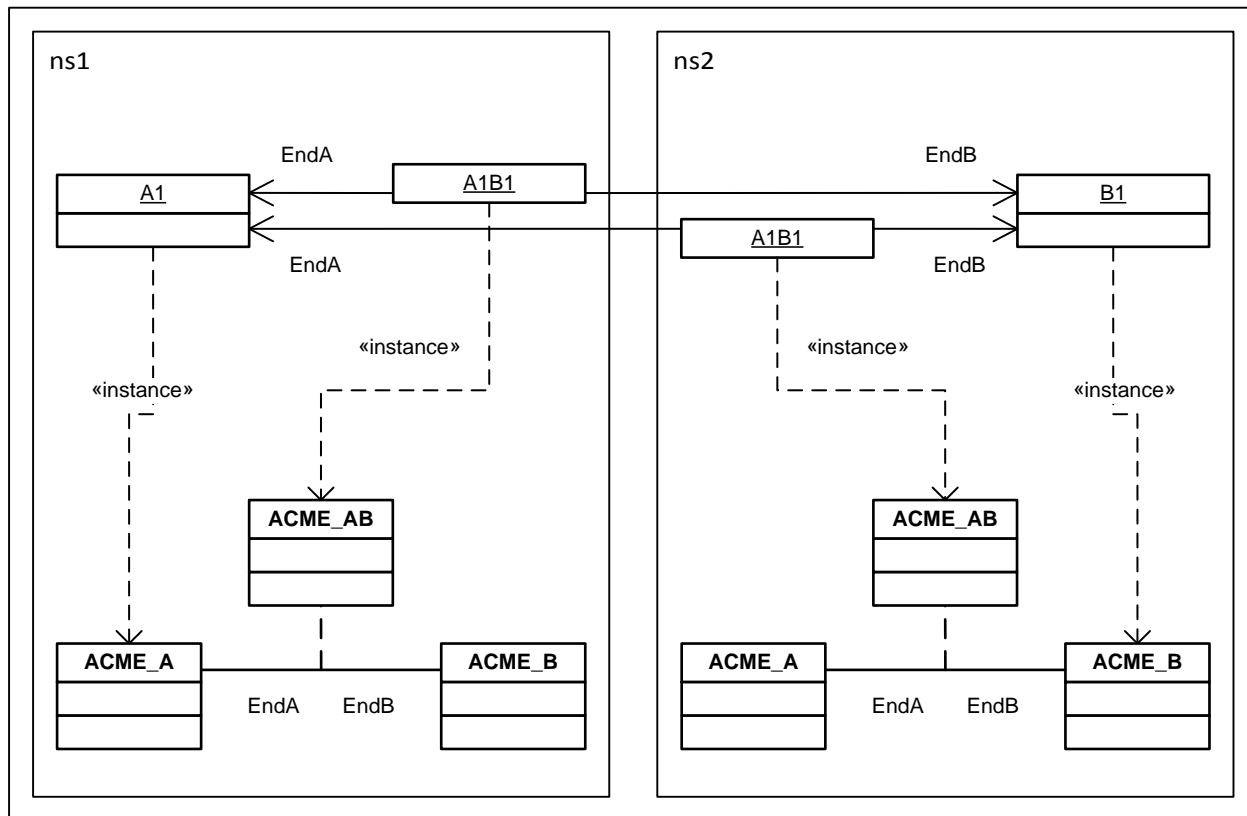
2473  
 2474

#### 2475 Figure 4 – Typical profile representation of binary association crossing namespaces

2476 The namespaces (in this scenario, `ns1` and `ns2`) are shown as boxes around a number of classes used  
 2477 in the management profile. Class `ACME_A` is in namespace `ns1`, class `ACME_B` in namespace `ns2`.  
 2478 Association `ACME_AB` crosses between these two namespaces and is therefore termed a *cross-*  
 2479 *namespace association*.

2480 This way of drawing the situation leaves it open exactly which classes and instances exist in each of the  
 2481 namespaces, and which of them are returned by instance-level and class-level association traversal  
 2482 operations.

2483 Figure 5 is a UML structure diagram showing the classes and instances in a WBEM server that need to  
 2484 exist when the classes shown in Figure 4 are implemented for bidirectional association traversal.



2485  
 2486

2487 **Figure 5 – Binary association: WBEM server objects for bidirectional traversal**

2488 The upper part of the figure shows instances, the lower part shows classes. Both of these are objects in a  
 2489 particular namespace of a WBEM server. Note that every object in this diagram is contained in a  
 2490 namespace. This is consistent with [DSP0004](#) which defines that the name of every object (class,  
 2491 instance, qualifier type) has a namespace path component. As a result, the instance A1B1 of the cross-  
 2492 namespace association ACME\_AB appears in each of the two namespaces; in a way, it is duplicated.

2493 **Rule:** Conformant implementations of bidirectional association traversal across namespaces shall have  
 2494 any such bidirectional cross-namespace association instances exist in both namespaces, and shall have  
 2495 the instances associated through such cross-namespace associations exist in only one namespace.

2496 Enumerating the instances of the association class ACME\_AB in namespace ns1 (e.g. with the  
 2497 EnumerateInstances operation) returns instance ns1::A1B1, and enumerating the instances of  
 2498 ACME\_AB in namespace ns2 returns instance ns2::A1B1. This means that the association instances act  
 2499 like any other instances: They can be enumerated (if the operation is implemented) and that enumeration  
 2500 is scoped to a particular namespace. The instances ns1::A1B1 and ns2::A1B1 are distinct instances,  
 2501 because their namespace path is different.

2502 The instances of the associated classes ACME\_A1 and ACME\_B1 appear only in their respective  
 2503 namespaces; they are not duplicated. As a result, enumerating the instances of ACME\_A in namespace  
 2504 ns1 returns ns1::A1, and enumerating the instances of ACME\_A in ns2 returns no instances (the  
 2505 EnumerateInstances operation still succeeds, because the class ACME\_A exists in ns2).



2506 The association traversal operations work in both directions in this scenario:

2507 Traversing association `ACME_AB` starting from instance `ns1::A1` using the `Associators` operation results  
2508 in instance `ns2::B1`, and traversing association `ACME_AB` starting from instance `ns2::B1` using the  
2509 `Associators` operation results in instance `ns1::A1`. Because this behavior can be determined from the  
2510 descriptions of these operations for the single-namespace case, no special rule for the cross-namespace  
2511 case has been defined.

2512 Because of the duplication of association instances in both namespaces, the situation is not intuitively  
2513 clear for the `References` operation and other association-returning operations: The association instances  
2514 `ns1::A1B1` and `ns2::A1B1` both reference the instance `ns1::A1`, so from a perspective of following  
2515 the specified behavior for this operation by the letter, one can argue that both instances need to be  
2516 returned, because they both exist and both reference the source instance. However, because these two  
2517 instances are logically the same, a client would need to reduce the result set by eliminating such logical  
2518 duplicates. Therefore, this annex defines the following restricting rule:

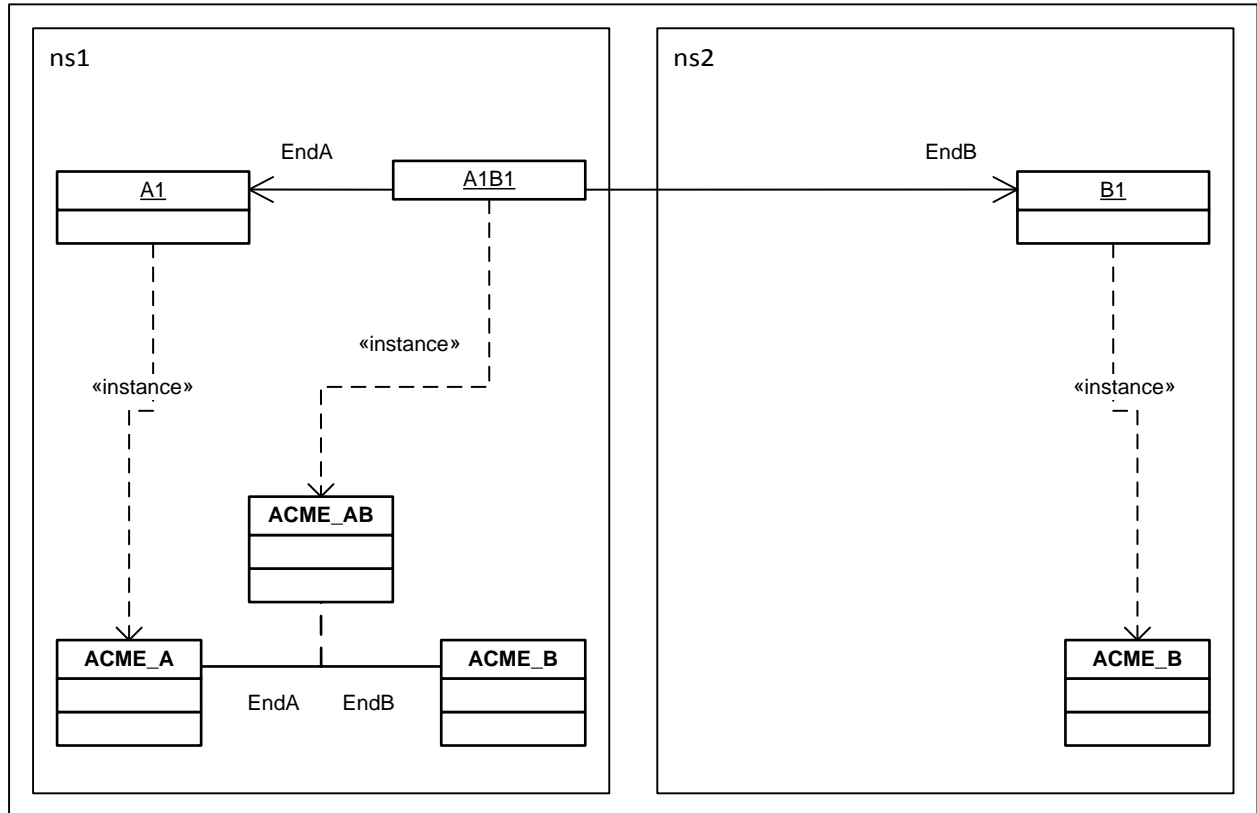
2519 **Rule:** Conformant implementations of the `References` operation and other association-returning  
2520 operations (such as `ReferenceNames` and `OpenReferences`) shall return only association instances that  
2521 exist in the namespace of the source instance (even if a duplicated association instance exists in the  
2522 other namespace).

2523 For classes, the existence requirements are driven by their role as a creation class, and their role as a  
2524 declared target of a reference in an association class. In Figure 5, the four classes that are the target of  
2525 the `<<instance>>` dependency need to exist because they have instances in their namespace.  
2526 Because the references in an association class only declare their targeted class but not their targeted  
2527 namespace, schema consistency rules require that all classes referenced by an association class exist as  
2528 objects in the same namespace as the association class. As a result, class `ACME_A` in addition needs to  
2529 exist in `ns2`, and class `ACME_B` in addition needs to exist in `ns1`. Note that this is driven by consistency  
2530 rules within a schema in a namespace, and is independent of whether or not class-level association  
2531 traversal operations are supported. As a result, no additional rule needs to be defined for the existence of  
2532 class objects in a cross-namespace case.

2533 Because of the limitation that class-level references do not declare a target namespace, this annex  
2534 defines the following rule for the behavior of the class-level operations:

2535 **Rule:** Conformant implementations of the `AssociatorClasses` and `ReferenceClasses` operation shall  
2536 return only classes that exist in the namespace of the source class; they never cross namespace  
2537 boundaries.

2538 Figure 6 shows the classes and instances in a WBEM server that need to exist when the classes shown  
2539 in Figure 4 are implemented for unidirectional association traversal in the direction from `ns1` to `ns2`:



2540  
2541

2542 **Figure 6 – Binary association: WBEM server objects for unidirectional traversal**

2543 In this case, the association instance `A1B1` only exists in namespace `ns1`, where traversal starts from.

2544 **Rule:** Conformant implementations of unidirectional association traversal across namespaces shall have  
 2545 any such unidirectional cross-namespace association instances exist in only the source namespace  
 2546 where traversal starts from, and shall have the instances associated through such cross-namespace  
 2547 associations exist in only one namespace.

2548 Because there is no instance `A1B1` in namespace `ns2`, there is no need for the classes `ACME_AB` and  
 2549 `ACME_A` to exist in `ns2`. As a result, namespace `ns2`, is "logically unaware" that namespace `ns1` can  
 2550 traverse into it. Whether this implies "implementation unawareness" depends on the type of WBEM server  
 2551 infrastructure that is used.

## ANNEX B (informative)

### Change log

2552  
2553  
2554  
2555

2556

Version	Date	Description
1.0.0	2010-04-22	<p>Published as DMTF Standard, with the following changes:</p> <ul style="list-style-type: none"> <li>• Consolidated terminology with DSP0004 2.6 and DSP1001 1.1.</li> <li>• Simplified the definition of generic types by relating them to DSP0004 2.6.</li> <li>• Clarifications for error handling and for pre- and postconditions.</li> <li>• Added definition of ACID properties and defined ACID requirements on all operations.</li> <li>• CreateInstance: Fixed incorrect statement about initial value if a property defines no default value in its class declaration.</li> <li>• ModifyClass: Removed message WIPG0232.</li> <li>• OpenQueryInstances: Removed message WIPG0124.</li> <li>• OpenAssociatedInstances...: Replaced message WIPG0214 with WIPG0213.</li> <li>• OpenReferencingInstances...: Replaced message WIPG0214 with WIPG0213.</li> <li>• GetAssociatedInstances...: Added message WIPG0213.</li> <li>• GetReferencingInstances...: Added message WIPG0213.</li> <li>• Removed ExecQuery operation and QueryResult type.</li> <li>• Removed GetAssociatedGraphInstancesWithPath and OpenAssociatedGraphInstancesWithPath and added operations for retrieval of associated instance graphs into ANNEX A (Future operations).</li> <li>• Stated the messages to be used for precondition violations. This affects all operations.</li> <li>• Added sources of messages (infrastructure / class implementation). This affects all operations.</li> <li>• Added usage of message WIPG0249 as needed and adjusted the name of message WIPG0208, to accommodate the DSP8016 change that splits message WIPG0208 into WIPG0208 and WIPG0249. This affects most operations.</li> <li>• Removed informative annex about required updates to other DMTF specifications.</li> <li>• Moved reference to DSP1001 into Bibliography</li> <li>• Changed terms: WBEM server, WBEM client, WBEM operation, WBEM protocol, WBEM listener, WBEM indication; Added references to document related terms in ISO guidelines.</li> <li>• Added "class implementation" as an additional source for error message WIPG0240 (WBEM server limits are exceeded) and WIPG0249 (Invalid input parameter value) on all instance related operations that use these messages</li> <li>• Generalized name of message WIPG0222 from "Query language feature not supported by WBEM server infrastructure" to "Query language feature not supported", following the corresponding change in DSP8016 1.0.1</li> <li>• Clarified that error message source (class specific vs. infrastructure) is a recommendation only</li> <li>• Changed DeleteDependents parameter of DeleteClass operation to be experimental</li> </ul>

Version	Date	Description
1.1.0	2015-01-07	<p>Published as DMTF Standard, with the following changes:</p> <ul style="list-style-type: none"> <li>• Fixed an error in the description of the IncludeInheritedElements parameter of the GetSubClassesWithPath operation (it is based on the specified class, not on the returned classes).</li> <li>• Clarified why the GetTopClassesWithPath operation does not have an IncludeInheritedElements parameter.</li> <li>• Deprecated the IncludeProperties parameter of the GetClass, GetAssociatedClassesWithPath, and GetReferencingClassesWithPath operations, with no replacement.</li> <li>• In CreateInstance, fixed that property default values are now treated as an initialization constraint, together with the PropertyConstraint qualifier and constraints defined in management profiles.</li> <li>• In GetClass, fixed the Class output parameter to now be of type ClassSpecification (it was ClassSpecificationWithPath, returning the input class path again).</li> <li>• Added support for indications.</li> <li>• Added IncludeInheritedElements parameter to getClass.</li> <li>• Clarified that class origin information indicates the leaf-most class defining the element, in override situations, consistent with DSP0200 1.4.</li> <li>• Narrowed the definition of "WBEM protocol" in that it needs to conform to generic operations.</li> <li>• Improved the description of the interaction model and distinction between generic operations and WBEM protocol.</li> <li>• Updated the minor versions of several normative references, added DSP1054 as a normative reference, and moved DSP0228 to the Bibliography.</li> <li>• Wording improvements throughout the document.</li> <li>• From 1.0.2: Errata: Changed the names of the generic operations to be aligned with the CIM-XML operation names. See ANNEX B for details.</li> <li>• From 1.0.2: For the PullInstances operation, fixed an incorrect occurrence of its name, and an error in its description where it was incorrectly stated that it would return instances with path.</li> <li>• Added references to CIM-RS specifications.</li> <li>• Added requirement for WBEM protocols to support FQL (Filter Query Language).</li> <li>• Deprecated non-pulled instance operations that have pulled equivalents (EnumerateInstances, EnumerateInstanceNames, Associators, AssociatorNames, References, ReferenceNames).</li> <li>• Deprecated pulled instance operations returning instance paths (OpenEnumerateInstancePaths, OpenAssociatorPaths, OpenReferencePaths).</li> <li>• Deprecated EnumerationCount operation.</li> <li>• Deprecated IncludeClassOrigin input parameter on any instance operations (GetInstance, EnumerateInstances, Associators, References, OpenEnumerateInstances, OpenAssociators, OpenReferences).</li> <li>• Clarified which components have to be present in any object paths (namespace, instance, class, qualifier type).</li> <li>• Clarified requirements for existence of classes and namespaces of any class-level and instance-level (pulled and non-pulled) association operations, by adding according preconditions.</li> <li>• Errata: Fixed the missing WIPG0214 (Class not found) in pulled association operations.</li> <li>• Errata: Changed the behavior of (pulled and non-pulled) association operations in case the source instance does not exist, from failing to succeeding with an empty result set, in order to be aligned with the behavior of the corresponding CIM-XML operations. As a result, removed WIPG0213 (Instance not found) from their set of allowable error messages.</li> <li>• Errata: Removed the AssociatedClassName and AssociatedRoleName filters from (pulled and non-pulled) reference operations, in order to be aligned with the filtering abilities of the corresponding CIM-XML operations.</li> </ul>

Version	Date	Description
		<ul style="list-style-type: none"> <li>• Errata: Changed the name of parameter RoleName to SourceRoleName, of the class-level association operations, for consistency with the corresponding instance-level operations.</li> <li>• Added WIPG0240 (WBEM server limits are exceeded) to all operations that did not have it yet.</li> <li>• Added WIPG0214 (Class not found) to InvokeMethod operation.</li> <li>• Added ANNEX C, defining normative rules for cross-namespace associations.</li> <li>• Terms and abbreviations are now based on <a href="#">DSP0198</a>.</li> </ul>
2.0.0	2015-03-06	<p>Published as DMTF Standard, with the following changes:</p> <ul style="list-style-type: none"> <li>• Removed the deprecated direct instance enumeration and association operations (EnumerateInstances, EnumerateInstanceNames, Associators, AssociatorNames, References, ReferenceNames).</li> <li>• Removed the deprecated pulled instance operations returning instance paths (OpenEnumerateInstancePaths, OpenAssociatorPaths, OpenReferencePaths and PullInstancePaths).</li> <li>• Removed the deprecated EnumerationCount operation.</li> <li>• Removed the class enumeration and association operations that return class names or class paths (EnumerateClassNames, AssociatorClassPaths, ReferenceClassPaths).</li> <li>• Removed the deprecated IncludeClassOrigin parameter from any instance operations.</li> <li>• Removed the ExcludeSubclassProperties parameter from any instance operations.</li> <li>• Removed the IncludeClassOrigin parameter from any class operations.</li> <li>• Removed the deprecated IncludedProperties parameter from any class operations.</li> <li>• Removed the annex about changed operation names.</li> <li>• Removed the annex about future ideas.</li> <li>• Removed alternative options for error handling, to require support for extended error handling using the generic error messages.</li> <li>• Clarified the meaning of requirement levels for generic error messages (see 5.7) and for operation parameters (see 5.3.3).</li> </ul>

2557

## Bibliography

2558

- 2559 DMTF DSP0200, *CIM Operations over HTTP 1.3*,  
2560 [http://www.dmtf.org/standards/published\\_documents/DSP0200\\_1.3.pdf](http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf)
- 2561 DMTF DSP0201, *Representation of CIM in XML 2.3*,  
2562 [http://www.dmtf.org/standards/published\\_documents/DSP0201\\_2.3.pdf](http://www.dmtf.org/standards/published_documents/DSP0201_2.3.pdf)
- 2563 DMTF DSP0202, *CIM Query Language Specification 1.0*,  
2564 [http://www.dmtf.org/standards/published\\_documents/DSP0202\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0202_1.0.pdf)
- 2565 DMTF DSP0203, *DTD for Representation of CIM in XML 2.3*,  
2566 [http://www.dmtf.org/standards/published\\_documents/DSP0203\\_2.3.dtd](http://www.dmtf.org/standards/published_documents/DSP0203_2.3.dtd)
- 2567 DMTF DSP0210, *CIM-RS Protocol 2.0*,  
2568 [http://www.dmtf.org/standards/published\\_documents/DSP0210\\_2.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0210_2.0.pdf)
- 2569 DMTF DSP0211, *CIM-RS Payload Representation in JSON 2.0*,  
2570 [http://www.dmtf.org/standards/published\\_documents/DSP0211\\_2.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0211_2.0.pdf)
- 2571 DMTF DSP0214, *Server Management Command Line Protocol Specification 1.0*,  
2572 [http://www.dmtf.org/standards/published\\_documents/DSP0214\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0214_1.0.pdf)
- 2573 DMTF DSP0226, *Web Services for Management 1.0*,  
2574 [http://www.dmtf.org/standards/published\\_documents/DSP0226\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0226_1.0.pdf)
- 2575 DMTF DSP0227, *WS-Management CIM Binding Specification 1.0*,  
2576 [http://www.dmtf.org/standards/published\\_documents/DSP0227\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0227_1.0.pdf)
- 2577 DMTF DSP0228, *Message Registry XML Schema 1.1*,  
2578 [http://schemas.dmtf.org/wbem/messageregistry/1/dsp0228\\_1.1.xsd](http://schemas.dmtf.org/wbem/messageregistry/1/dsp0228_1.1.xsd)
- 2579 DMTF DSP0230, *WS-CIM Mapping Specification 1.0*,  
2580 [http://www.dmtf.org/standards/published\\_documents/DSP0230\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0230_1.0.pdf)
- 2581 DMTF DSP1001, *Management Profile Specification Usage Guide 1.2*,  
2582 [http://www.dmtf.org/standards/published\\_documents/DSP1001\\_1.2.pdf](http://www.dmtf.org/standards/published_documents/DSP1001_1.2.pdf)
- 2583 DMTF DSP8028, *Management Profile XML Schema 1.1*,  
2584 [http://schemas.dmtf.org/wbem/mgmtprofile/1/dsp8028\\_1.1.xsd](http://schemas.dmtf.org/wbem/mgmtprofile/1/dsp8028_1.1.xsd)
- 2585 JCP JSR-48, *Java Community Process JSR-48: WBEM servers Specification*, not yet published,  
2586 <http://jcp.org/en/jsr/detail?id=48>
- 2587 The Open Group CMPI, *Systems Management: Common Manageability Programming Interface (CMPI)*,  
2588 *Issue 2.0*, <http://www.opengroup.org/bookstore/catalog/c061.htm>