



1
2
3
4

Document Number: DSP-IS0201

Date: 2011-04-15

Version: 1.0.0

5 **CIM Operations Over RESTful Services**

- 6 **Document Type: Specification**
- 7 **Document Status: DMTF Informational Specification**
- 8 **Document Language: en-US**

9 IMPORTANT: This specification is not a standard. It is an exploratory, informational document developed
10 in order to obtain industry feedback. It does not reflect the views of the DMTF or all of its members. It is
11 possible that future standards may or may not consider this work product to be an input in whole or in
12 part.

13

14 Copyright Notice

15 Copyright © 2010,2011 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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30 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

CONTENTS

37	Foreword	8
38	Introduction	9
39	Document conventions.....	9
40	Typographical conventions	9
41	ABNF usage conventions	9
42	Experimental material	9
43	1 Scope	11
44	2 Normative references.....	11
45	3 Terms and definitions	12
46	4 Symbols and abbreviated terms.....	15
47	5 Concepts	17
48	5.1 REST architectural style supported by CIM-RS	17
49	6 Conformance.....	18
50	7 REST resources and their resource identifiers	19
51	7.1 Structure of WBEM resource identifiers.....	19
52	7.1.1 General structure	19
53	7.1.2 Percent-encoding.....	20
54	7.1.3 Scheme.....	22
55	7.1.4 Authority.....	22
56	7.1.5 Client knowledge about URI structure	22
57	7.2 REST resources and their URIs	22
58	7.2.1 Namespace collection resource.....	23
59	7.2.2 Namespace resource.....	23
60	7.2.3 Class collection resource.....	23
61	7.2.4 Class resource.....	23
62	7.2.5 Class associator collection resource	24
63	7.2.6 Class reference collection resource.....	24
64	7.2.7 Class method invocation resource.....	24
65	7.2.8 Instance collection resource	24
66	7.2.9 Instance resource	24
67	7.2.10 Instance associator collection resource.....	25
68	7.2.11 Instance reference collection resource	25
69	7.2.12 Instance method invocation resource	25
70	7.2.13 Qualifier type collection resource.....	25
71	7.2.14 Qualifier type resource.....	25
72	7.2.15 Instance query resource	25
73	7.3 Query parameters in URIs	26
74	7.3.1 acn (associated class name)	26
75	7.3.2 arn (associated role name)	27
76	7.3.3 c (class).....	27
77	7.3.4 dd (delete dependents).....	28
78	7.3.5 esbp (exclude subclass properties)	28
79	7.3.6 fql (filter query language)	29
80	7.3.7 fqs (filter query string)	29
81	7.3.8 ico (include class origin)	30
82	7.3.9 iie (include inherited elements)	31
83	7.3.10 ip (included properties)	31
84	7.3.11 iq (include qualifiers).....	32
85	7.3.12 isbc (include subclasses).....	32
86	7.3.13 n (namespace).....	33
87	7.3.14 pr (paged retrieval).....	33

88	7.3.15	q (qualifier)	34
89	7.3.16	rcn (referencing class name)	35
90	7.3.17	spc (superclass).....	35
91	7.3.18	srn (source role name).....	36
92	8	Operations	36
93	8.1	Overview on REST resources and HTTP methods	36
94	8.2	Common behaviors for all operations	37
95	8.2.1	Content negotiation.....	37
96	8.2.2	Links.....	38
97	8.2.3	Verifying the basis of resource modifications	39
98	8.2.4	Caching of responses	39
99	8.2.5	Success and failure	40
100	8.2.6	Error messages.....	40
101	8.2.7	Consistency model.....	40
102	8.2.8	Common operation parameters for all operations	40
103	8.3	Optional behaviors for CIM-RS protocol.....	40
104	8.3.1	Entity tagging feature	40
105	8.3.2	Paged retrieval feature	40
106	8.4	Protocol payload elements	42
107	8.4.1	NamespaceCollection payload element	42
108	8.4.2	Namespace payload element	43
109	8.4.3	ClassCollection payload element.....	43
110	8.4.4	Class payload element	44
111	8.4.5	InstanceCollection payload element.....	44
112	8.4.6	Instance payload element.....	45
113	8.4.7	QualifierTypeCollection payload element	46
114	8.4.8	QualifierType payload element	46
115	8.4.9	MethodInvocationRequest payload element	47
116	8.4.10	MethodInvocationResponse payload element.....	47
117	8.4.11	InstanceModificationRequest payload element	48
118	8.4.12	InstanceQueryRequest payload element	48
119	8.4.13	ErrorResponse payload element	49
120	8.5	WBEM server and listener operations	49
121	8.5.1	OPTIONS.....	49
122	8.6	Namespace collection operations.....	50
123	8.6.1	GET namespace collection.....	50
124	8.7	Namespace operations.....	52
125	8.7.1	GET namespace	52
126	8.8	Class collection operations	53
127	8.8.1	GET class collection	53
128	8.8.2	POST class collection	56
129	8.9	Class operations	57
130	8.9.1	GET class	57
131	8.9.2	DELETE class.....	58
132	8.9.3	PUT class.....	59
133	8.10	Class associator collection operations.....	59
134	8.10.1	GET class associator collection.....	59
135	8.11	Class reference collection operations.....	61
136	8.11.1	GET class reference collection	61
137	8.12	Class method operations	63
138	8.12.1	POST class method	63
139	8.13	Instance collection operations	64
140	8.13.1	GET instance collection	64
141	8.13.2	POST instance collection.....	66
142	8.14	Instance operations.....	67
143	8.14.1	GET instance	67

144	8.14.2	DELETE instance.....	68
145	8.14.3	PUT instance	69
146	8.14.4	PATCH instance	70
147	8.15	Instance associator collection operations.....	72
148	8.15.1	GET instance associator collection.....	72
149	8.16	Instance reference collection operations	74
150	8.16.1	GET instance reference collection.....	74
151	8.17	Instance method operations	76
152	8.17.1	POST instance method.....	76
153	8.18	Qualifier type collection operations.....	77
154	8.18.1	GET qualifier type collection	77
155	8.18.2	POST qualifier type collection.....	78
156	8.19	Qualifier type operations.....	79
157	8.19.1	GET qualifier type	79
158	8.19.2	DELETE qualifier type	80
159	8.19.3	PUT qualifier type	81
160	8.20	Instance query operations	82
161	8.20.1	POST instance query.....	82
162	8.21	Functionality without specific operations	83
163	8.21.1	Subscription and other indication related functions.....	83
164	8.22	Potential future operations.....	83
165	8.22.1	Indication delivery	83
166	9	Usage of HTTP and HTTPS.....	84
167	9.1	HTTP and HTTPS version requirements	84
168	9.2	Authentication requirements	84
169	9.2.1	Operating without authentication	84
170	9.2.2	HTTP basic authentication.....	84
171	9.2.3	HTTP digest authentication	85
172	9.2.4	Other authentication mechanisms	85
173	9.3	Message encryption.....	85
174	9.4	HTTP header fields	85
175	9.4.1	Accept.....	85
176	9.4.2	Content-Type	86
177	9.4.3	Etag.....	86
178	9.4.4	If-Match.....	87
179	9.4.5	CIMRS-Content-Types	87
180	9.4.6	CIMRS-Entity-Tagging-Feature	87
181	9.4.7	CIMRS-Paged-Retrieval-Feature.....	88
182	9.4.8	CIMRS-Filter-Query-Languages.....	88
183	9.4.9	CIMRS-Instance-Query-Languages	88
184	10	Payload representation	89
185	10.1	Media type	89
186	10.2	Payload element representations	89
187	ANNEX A (informative)	Known payload representations	91
188	ANNEX B (informative)	Examples for structure of resource URIs	92
189	B.1	Example using CIM object types as URI segments.....	92
190	B.1.1	Namespace collection resource.....	92
191	B.1.2	Namespace resource.....	92
192	B.1.3	Class collection resource.....	93
193	B.1.4	Class resource.....	93
194	B.1.5	Class associator collection resource	93
195	B.1.6	Class reference collection resource.....	94
196	B.1.7	Class method invocation resource.....	94
197	B.1.8	Instance collection resource	94
198	B.1.9	Instance resource	94

199	B.1.10 Instance associator collection resource.....	96
200	B.1.11 Instance reference collection resource.....	96
201	B.1.12 Instance method invocation resource.....	96
202	B.1.13 Qualifier type collection resource.....	97
203	B.1.14 Qualifier type resource.....	97
204	B.1.15 Instance query resource	97
205	B.2 Example using WBEM URIs	98
206	ANNEX C (informative) Change log.....	99
207	Bibliography	100
208		

209 Tables

210	Table 1 – REST resources and HTTP methods	36
211	Table 2 – Links included in payload elements	38
212	Table 3 – Operations supporting paged retrieval.....	41
213	Table 4 – Additional links for paged retrieval.....	42
214	Table 5 – Properties of NamespaceCollection payload element.....	42
215	Table 6 – Links of NamespaceCollection payload element.....	42
216	Table 7 – Properties of Namespace payload element.....	43
217	Table 8 – Links of Namespace payload element.....	43
218	Table 9 – Properties of ClassCollection payload element	43
219	Table 10 – Links of ClassCollection payload element	43
220	Table 11 – Properties of Class payload payload element	44
221	Table 12 – Links of Class payload element	44
222	Table 13 – Properties of InstanceCollection payload element.....	45
223	Table 14 – Links of InstanceCollection payload element.....	45
224	Table 15 – Properties of Instance payload element	45
225	Table 16 – Links of Instance payload element.....	46
226	Table 17 – Properties of QualifierTypeCollection payload element.....	46
227	Table 18 – Links of QualifierTypeCollection payload element.....	46
228	Table 19 – Properties of QualifierType payload element.....	47
229	Table 20 – Links of QualifierType payload element.....	47
230	Table 21 – Properties of MethodInvocationRequest payload element.....	47
231	Table 22 – Properties of MethodInvocationResponse payload element	48
232	Table 23 – Properties of InstanceModificationRequest payload element.....	48
233	Table 24 – Properties of PropertyValue generic type	48
234	Table 25 – Properties of InstanceQueryRequest payload element	49
235	Table 26 – Properties of ErrorResponse payload element.....	49
236	Table 27 – Response header fields for OPTIONS to WBEM server	50
237	Table 28 – Response header fields for OPTIONS to WBEM listener.....	50
238	Table 29 – Query parameters for GET namespace collection.....	51
239	Table 30 – Query parameters for GET namespace.....	53
240	Table 31 – Query parameters for GET class collection	54
241	Table 32 – Query parameters for GET class	57
242	Table 33 – Query parameters for DELETE class	58
243	Table 34 – Query parameters for GET class associator collection.....	60
244	Table 35 – Query parameters for GET class reference collection.....	62

245	Table 36 – Query parameters for GET instance collection.....	65
246	Table 37 – Query parameters for GET instance.....	67
247	Table 38 – Query parameters for GET instance associator collection	72
248	Table 39 – Query parameters for GET instance reference collection	74
249	Table 40 – Query parameters for GET qualifier type collection.....	77
250	Table 41 – Known CIM-RS payload representations.....	91
251		

252

Foreword

253 The *CIM Operations Over RESTful Services* (DSP-IS0201) informational specification was prepared by
254 the DMTF CIM-RS Incubator.

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

257 **Acknowledgments**

258 The DMTF acknowledges the following individuals for their contributions to this document:

- 259 • Andreas Maier, IBM (editor)
- 260 • Cornelia Davis, EMC
- 261 • George Ericson, EMC
- 262 • Johannes Holzer, IBM
- 263 • Robert Kieninger, IBM
- 264 • Larry Lamers, VMware
- 265 • Marvin Waschke, Computer Associates

266

Introduction

267 The information in this document should be sufficient for a WBEM server, WBEM client, and WBEM
 268 listener to unambiguously identify the protocol interactions that shall be performed to implement the CIM-
 269 RS protocol.

270 NOTE: This version of this document describes the WBEM listener and WBEM server in their roles related to
 271 indication delivery, but it does not yet describe the indication delivery as a protocol interaction.
 272

273 The target audience for this specification is implementers who are writing WBEM servers, WBEM clients,
 274 and WBEM listeners supporting the CIM-RS protocol.
 275

276 Document conventions

277 Typographical conventions

278 The following typographical conventions are used in this document:

- 279 • Document titles are marked in *italics*.
- 280 • Important terms that are used for the first time are marked in *italics*.
- 281 • Terms include a link to the term definition in the "Terms and definitions" clause, enabling easy
 282 navigation to the term definition.
- 283 • ABNF rules are in `monospaced font`.

284 ABNF usage conventions

285 Format definitions in this document are specified using ABNF (see [RFC5234](#)), with the following
 286 deviations:

- 287 • Literal strings are to be interpreted as case-sensitive Unicode characters, as opposed to the
 288 definition in [RFC5234](#) that interprets literal strings as case-insensitive US-ASCII characters.

289 The following ABNF rules may be used in any ABNF in this document:

290 `WS = *(U+0020 / U+0009 / U+000A) ; zero or more white space characters`

291 Experimental material

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

297 The following typographical convention indicates experimental material:

298 **EXPERIMENTAL**

299 Experimental material appears here.

300 **EXPERIMENTAL**

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

303

CIM Operations Over RESTful Services

304 1 Scope

305 This informational specification describes the *CIM Operations Over RESTful Services* protocol (also
306 referred to as the *CIM-RS* protocol), which defines the use of RESTful services as a communications
307 protocol between WBEM client, WBEM server, and WBEM listener.

308 The semantic of the operations defined for the CIM-RS protocol conforms to the generic operations
309 defined in [DSP0223](#). Because the current version of [DSP0223](#) does not yet cover indications, this version
310 of this document also does not support indications.

311 The CIM-RS protocol can be used with HTTP and HTTPS.

312 The CIM-RS protocol supports multiple payload representations. These payload representations are
313 expected to be described in separate documents. Their use within the CIM-RS protocol is determined
314 through HTTP content negotiation.

315 2 Normative references

316 The following referenced documents are indispensable for the application of this document. For dated or
317 versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies.
318 For references without a date or version, the latest published edition of the referenced document
319 (including any corrigenda or DMTF update versions) applies.
320

321 DMTF DSP0004, *CIM Infrastructure Specification 2.6*,
322 http://www.dmtf.org/standards/published_documents/DSP0004_2.6.pdf

323 DMTF DSP0223, *Generic Operations 1.0*,
324 http://www.dmtf.org/standards/published_documents/DSP0223_1.0.pdf

325 DMTF DSP0228, *Message Registry XML Schema 1.1*,
326 http://schemas.dmtf.org/wbem/messageregistry/1/dsp0228_1.1.xsd

327 DMTF DSP1033, *Profile Registration Profile 1.0*,
328 http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf

329 DMTF DSP8016, *WBEM Operations Message Registry 1.0*,
330 http://schemas.dmtf.org/wbem/messageregistry/1/dsp8016_1.0.xml

331 ECMA-262, *ECMAScript Language Specification, 5th Edition, December 2009*,
332 <http://www.ecma-international.org/publications/standards/Ecma-262.htm>

333 IETF RFC2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999,
334 <http://tools.ietf.org/html/rfc2616>

335 IETF RFC2617, *HTTP Authentication: Basic and Digest Access Authentication*, June 1999,
336 <http://tools.ietf.org/html/rfc2617>

- 337 IETF RFC2818, *HTTP Over TLS*, May 2000,
338 <http://tools.ietf.org/html/rfc2818>
- 339 IETF RFC3986, *Uniform Resource Identifier (URI): Generic Syntax*, January 2005,
340 <http://tools.ietf.org/html/rfc3986>
- 341 IETF RFC4346, *The Transport Layer Security (TLS) Protocol, Version 1.1*, April 2006,
342 <http://tools.ietf.org/html/rfc4346>
- 343 IETF RFC4648, *The Base16, Base32, and Base64 Data Encodings*, October 2006,
344 <http://tools.ietf.org/html/rfc4648>
- 345 IETF RFC5005, *Feed Paging and Archiving*, September 2007,
346 <http://tools.ietf.org/html/rfc5005>
- 347 IETF RFC5234, *Augmented BNF for Syntax Specifications: ABNF*, January 2008,
348 <http://tools.ietf.org/html/rfc5234>
- 349 IETF RFC5246, *The Transport Layer Security (TLS) Protocol, Version 1.2*, August 2008,
350 <http://tools.ietf.org/html/rfc5246>
- 351 IETF RFC5789, *PATCH Method for HTTP*, March 2010,
352 <http://tools.ietf.org/html/rfc5789>
- 353 ISO/IEC 10646:2003, *Information technology -- Universal Multiple-Octet Coded Character Set (UCS)*,
354 [http://standards.iso.org/ittf/PubliclyAvailableStandards/c039921_ISO_IEC_10646_2003\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c039921_ISO_IEC_10646_2003(E).zip)
- 355 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards (2004, 5th*
356 *edition)*,
357 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse>
- 358 The Unicode Consortium, *The Unicode Standard, Version 5.2.0, Annex #15: Unicode Normalization*
359 *Forms*,
360 <http://www.unicode.org/reports/tr15/>

361 **3 Terms and definitions**

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

364 The terms "shall" ("required"), "shall not," "should" ("recommended"), "should not" ("not recommended"),
365 "may," "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described
366 in [ISO/IEC Directives, Part 2](#), Annex H. The terms in parenthesis are alternatives for the preceding term,
367 for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that
368 [ISO/IEC Directives, Part 2](#), Annex H specifies additional alternatives. Occurrences of such additional
369 alternatives shall be interpreted in their normal English meaning.

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

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

375 The terms defined in [DSP0004](#) and [DSP0223](#) apply to this document. The following additional terms are
376 used in this document.

377 **3.1**

378 **HTTP basic authentication**

379 a simple authentication scheme for use by HTTP and HTTPS that is based on providing credentials in
380 HTTP header fields. It is defined in [RFC2617](#).

381 **3.2**

382 **HTTP content negotiation**

383 a method for selecting a representation of content in an HTTP response when there are multiple
384 representations available. It is defined in section 12 of [RFC2616](#). Its use in the CIM-RS protocol is
385 described in 8.2.1.

386 **3.3**

387 **HTTP digest authentication**

388 an authentication scheme for use by HTTP and HTTPS that is based on verifying shared secrets that are
389 not exchanged. It is defined in [RFC2617](#).

390 **3.4**

391 **HTTP entity body**

392 the payload within an HTTP message, as defined in section 7.2 of [RFC2616](#)

393 **3.5**

394 **HTTP entity header**

395 an HTTP header field used in both HTTP request and/or HTTP response. Also called "entity header field".

396 **3.6**

397 **HTTP header field**

398 a named value used in the header of HTTP messages, as defined in section 4.2 of [RFC2616](#). Also called
399 "HTTP header".

400 **3.7**

401 **HTTP message**

402 an interaction between an HTTP client and an HTTP server (in any direction), as defined in section 4 of
403 [RFC2616](#)

404 **3.8**

405 **HTTP method**

406 the type of interaction stated in an HTTP request, as defined in section 5.1.1 of [RFC2616](#)

407 **3.9**

408 **HTTP request**

409 an HTTP message sent from an HTTP client to an HTTP server as defined in section 5 of [RFC2616](#). Also
410 called "HTTP request message".

411 **3.10**

412 **HTTP request header**

413 an HTTP header field used in an HTTP request. Also called "request header field".

414 **3.11**

415 **HTTP response**

416 an HTTP message sent from an HTTP server to an HTTP client, as defined in section 6 of [RFC2616](#). Also
417 called "HTTP response message".

- 418 **3.12**
419 **HTTP response header**
420 an HTTP header field used in an HTTP response. Also called "response header field".
- 421 **3.13**
422 **interop namespace**
423 a namespace containing CIM instances representing specific capabilities of a WBEM server, such as for
424 example, CIM_Namespace instances representing the namespaces of the WBEM server. For details, see
425 [DSP1033](#).
- 426 **3.14**
427 **link**
428 a WBEM resource identifier that appears in an HTTP entity body. For details, see 8.2.2.
- 429 **3.15**
430 **MIME media type**
431 a string identification for media types in Internet protocols, as defined in [RFC2045](#) and [RFC2046](#)
- 432 **3.16**
433 **Normalization Form C**
434 a normalization form for UCS characters that avoids the use of combining marks where possible and that
435 allows comparing UCS character strings on a per-code-point basis. It is defined in [The Unicode Standard](#),
436 [Annex #15](#).
- 437 **3.17**
438 **REST architectural style**
439 the architectural style described in [Architectural Styles and the Design of Network-based Software](#)
440 [Architectures](#), chapter 5, and in [REST APIs must be hypertext driven](#)
- 441 **3.18**
442 **REST resource**
443 any entity that is accessed through and targeted by a RESTful service
- 444 **3.19**
445 **RESTful service**
446 a service accessible using REST concepts
- 447 **3.20**
448 **UCS character**
449 a character from the Universal Character Set defined in [ISO/IEC 10646:2003](#)
- 450 **3.21**
451 **UCS code position**
452 a numeric identification for a UCS character in the range of 0x0 to 0x10FFFF, as defined in [ISO/IEC](#)
453 [10646:2003](#). See also [DSP0004](#) for an overview.
- 454 **3.22**
455 **Unicode character**
456 an alternate term for UCS character
- 457 **3.23**
458 **WBEM client**
459 a CIM client (see [DSP0004](#)) that supports a WBEM protocol. A WBEM client originates WBEM operations
460 for processing by a WBEM server. For details, see [DSP0223](#).

- 461 **3.24**
462 **WBEM indication**
463 an interaction within a WBEM protocol that is originated on a WBEM server and processed by a WBEM
464 listener. For details, see [DSP0223](#). This release of this document does not cover WBEM indications.
- 465 **3.25**
466 **WBEM listener**
467 a CIM listener (see [DSP0004](#)) that supports a WBEM protocol. A WBEM listener processes WBEM
468 indications originated by a WBEM server. For details, see [DSP0223](#).
- 469 **3.26**
470 **WBEM operation**
471 an interaction within a WBEM protocol that is originated by a WBEM client and processed by a WBEM
472 server. For details, see [DSP0223](#).
- 473 **3.27**
474 **WBEM protocol**
475 a communications protocol between WBEM client, WBEM server, and WBEM listener. A WBEM protocol
476 defines how the WBEM operations and WBEM indications work on top of an underlying protocol layer (for
477 example, HTTP, SOAP, or TCP). For details, see [DSP0223](#).
- 478 **3.28**
479 **WBEM resource identifier**
480 a URI that addresses a REST resource in a WBEM server. Some but not all of those REST resources
481 correspond to CIM objects in a WBEM server. Also called "resource identifier" in this document.
- 482 **3.29**
483 **WBEM server**
484 a CIM server (see [DSP0004](#)) that supports a WBEM protocol. A WBEM server processes WBEM
485 operations originated by a WBEM client and originates WBEM indications for processing by a WBEM
486 listener. For details, see [DSP0223](#).

487 **4 Symbols and abbreviated terms**

488 The abbreviations defined in [DSP0004](#) and [DSP0223](#) apply to this document. The following additional
489 abbreviations are used in this document.

- 490 **4.1**
491 **ABNF**
492 Augmented Backus-Naur Form, as defined in [RFC5234](#)
- 493 **4.2**
494 **CIM**
495 Common Information Model, as defined by DMTF
- 496 **4.3**
497 **CIM-RS**
498 CIM Operations Over RESTful Services; the name of the protocol defined in this document
- 499 **4.4**
500 **CQL**
501 CIM Query Language, as defined in [DSP0202](#)

- 502 **4.5**
503 **ECMAScript**
504 a scripting language that is the standard version of what was called JavaScript. It is defined in [ECMA-](#)
505 [262](#).
- 506 **4.6**
507 **HTTP**
508 Hyper Text Transfer Protocol. HTTP version 1.1 is defined in [RFC2616](#). Unless otherwise noted, the term
509 HTTP is used in this document to mean both HTTP and HTTPS.
- 510 **4.7**
511 **HTTPS**
512 Hyper Text Transfer Protocol Secure, as defined in [RFC2818](#)
- 513 **4.8**
514 **IANA**
515 Internet Assigned Numbers Authority; see <http://www.iana.org>
- 516 **4.9**
517 **JSON**
518 JavaScript Object Notation, as defined in [ECMA-262](#)
- 519 **4.10**
520 **MIME**
521 Multipurpose Internet Mail Extensions, as defined in [IANA MIME Media Types](#)
- 522 **4.11**
523 **REST**
524 Representational State Transfer, as originally and informally described in [Architectural Styles and the](#)
525 [Design of Network-based Software Architectures](#)
- 526 **4.12**
527 **UCS**
528 Universal Character Set, as defined in [ISO/IEC 10646:2003](#)
- 529 **4.13**
530 **URI**
531 Uniform Resource Identifier, as defined in [RFC3986](#)
- 532 **4.14**
533 **UTF-8**
534 UCS Transformation Format 8, as defined in [ISO/IEC 10646:2003](#)
- 535 **4.15**
536 **WBEM**
537 Web Based Enterprise Management, as defined by DMTF
- 538 **4.16**
539 **XML**
540 eXtensible Markup Language, as defined by W3C

541 5 Concepts

542 This clause defines concepts that are the basis for the definition of the CIM-RS protocol.

543 This document relies on the generic operations defined in [DSP0223](#) for a definition on the semantics of
544 the operations within the CIM-RS protocol. Thus, this document defines a mapping of generic operations
545 to HTTP messages, and of the input and output parameters of generic operations to resource identifiers,
546 HTTP header fields, and HTTP entity bodies.

547 5.1 REST architectural style supported by CIM-RS

548 CIM-RS follows most of the principles and constraints of the *REST architectural style* described by Roy
549 Fielding in chapter 5 of [Architectural Styles and the Design of Network-based Software Architectures](#) and
550 in [REST APIs must be hypertext driven](#). CIM-RS deviates from some of these principles and constraints,
551 as described in this clause.

552 The constraints defined in the REST architectural style are satisfied by CIM-RS as follows:

- 553 • **Client-Server:** The participants in CIM-RS have a client-server relationship between WBEM
554 client and WBEM server, and between WBEM server and WBEM listener. This constraint is fully
555 satisfied.
- 556 • **Stateless:** Interactions in CIM-RS are self-describing in that the server (that is, the WBEM
557 server in its server role, and the WBEM listener) does not maintain any session state. This
558 constraint is fully satisfied.

559 NOTE: Pulled enumeration operations as defined in [DSP0223](#) maintain the enumeration state either on
560 the server side or on the client side. In both approaches, the client needs to hand back and forth an
561 opaque data item called enumeration context, which is the actual enumeration state in case of a client-
562 maintained enumeration state, or a handle to the enumeration state in case of a server-maintained
563 enumeration state. CIM-RS supports both of these approaches. It is possible for a server to remain
564 stateless as far as the enumeration state goes, by implementing the client-based approach. The approach
565 implemented by a server is not visible to a client, because the enumeration context handed back and forth
566 is opaque to the client in both approaches.

- 567 • **Cache:** The HTTP methods used by CIM-RS are used as defined in [RFC2616](#) and [RFC5789](#)
568 (for PATCH). As a result, they are cacheable as defined in [RFC2616](#). This constraint is fully
569 satisfied.

570 NOTE: [RFC2616](#) defines only the result of HTTP GET methods to be cacheable.

- 571 • **Uniform interface:** The resources represented in CIM-RS are CIM namespaces, CIM classes,
572 CIM qualifier types, and CIM instances. CIM-RS defines a uniform interface for creating,
573 deleting, retrieving, replacing, and modifying these resources, based on HTTP methods. The
574 resource identifiers used in that interface are uniformly structured. This constraint is satisfied,
575 with the following deviation:

576 CIM methods can be invoked in CIM-RS through the use of HTTP POST. This may be seen as
577 a deviation from the REST architectural style, which suggests that any "method" be represented
578 as a modification of a resource. However, that is not practical in CIM, because significant effort
579 has been put into the definition of CIM method semantics in the CIM Schema and in
580 management profiles, and into existing implementations of these methods.

- 581 • **Layered system:** Layering is an inherent part of CIM, because it defines a CIM model of
582 managed objects in a managed environment and thus restricts knowledge of a client to only the
583 modeled representation of the managed environment. CIM-RS fully supports layering in that it
584 represents only the entities modeled in CIM. In addition, CIM-RS does not prevent the use of
585 HTTP intermediaries (for example, caches and proxy servers). This constraint is fully satisfied.
- 586 • **Code-On-Demand:** CIM-RS does not provide for sending any code to the client. This
587 constraint is not satisfied.

588 The representation of REST resources in CIM-RS is as follows:

- 589 • For CIM instances, its instance specification, including property values
- 590 • For CIM classes, its class specification, including any specified CIM qualifiers
- 591 • For CIM qualifier types, its qualifier type specification
- 592 • For CIM namespaces, the CIM instances of the CIM_Namespace class representing the
- 593 namespaces

594 This approach makes the nature and state of these REST resources available to a client. Because these
595 resource representations are defined at the level of the CIM model, the raw structure of any managed
596 objects represented in the CIM model is not directly visible at the interface. This pattern allows describing
597 REST resources independent of their specific managed environment platform or implementation.

598 The REST architectural style allows for the representation of static entities such as disk drives, or entities
599 with varying property values over time, such as a metric measuring the amount of available disk space at
600 a specific point in time, or entities that dynamically come into existence or cease to exist, such as a file
601 system mount. CIM-RS represents CIM modeled entities as REST resources, and thus can represent all
602 these types of entities.

603 In CIM-RS, resources are addressed through resource identifiers that are URIs. More precisely, resource
604 identifiers are used to address the CIM entities represented as REST resources. The REST architectural
605 style recommends that all addressing information for a resource is in the resource identifier (and not, for
606 example, in the HTTP header). In addition, it recommends that resource identifiers are opaque to clients
607 and clients should not be required to understand the structure of resource identifiers or be required to
608 assemble any resource identifiers. CIM-RS follows the recommendations that all addressing information
609 for a resource is in the resource identifier and on opaqueness and non-assembly of the resource
610 identifier.

611 The REST architectural style promotes late binding between the abstracted resource that is addressed
612 through a resource identifier and the resource representation that is chosen in the interaction between
613 client and server. CIM-RS follows this by supporting multiple types of resource representations that are
614 chosen through HTTP content negotiation. (For details, see 8.2.1.)

615 CIM-RS supports retrieval of parts of CIM classes or instances. These parts are selected through query
616 parameters in the resource identifier URI. That renders these parts to be separate REST resources (that
617 is, separate from the REST resource representing the entire CIM class or instance), following the
618 principles in the REST architectural style.

619 The only top-level resources a client needs to know in CIM-RS are the resource identifier URIs of any
620 CIM namespaces the client wants to interact with. From that point on, CIM-RS operations allow retrieval
621 of the resource identifiers of any further REST resources representing CIM classes, instances, and
622 qualifier types, by means of links returned along with previously returned resources.

623 **6 Conformance**

624 This clause defines the criteria for WBEM clients, WBEM servers, and WBEM listeners to implement the
625 CIM-RS protocol conformant to this document.

626 WBEM clients, WBEM servers, and WBEM listeners implement the CIM-RS protocol conformant to this
627 document only if they satisfy all provisions set out in this document.

628 The terms WBEM client, WBEM server, and WBEM listener in this document refer to WBEM clients,
629 WBEM servers, and WBEM listeners that are conformant to this document, without explicitly mentioning
630 that.

631 7 REST resources and their resource identifiers

632 This clause defines the REST resources used in the CIM-RS protocol and their resource identifiers.

633 These resource identifiers are called *WBEM resource identifiers* (or short: "resource identifier") in this
 634 document and are URIs that address REST resources accessible through a WBEM server. Some of
 635 these REST resources represent CIM objects accessible through the WBEM server, and some others are
 636 used to give additional context to HTTP methods such as GET.

637 The following example shows a resource identifier that addresses the CIM_ManagedElement class in the
 638 "root/cimv2" namespace accessible through the WBEM server at port 5988 of host "acme.com", using the
 639 "http" scheme:

```
640 http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_ManagedElement
```

641 7.1 Structure of WBEM resource identifiers

642 This clause defines the structure of WBEM resource identifiers.

643 7.1.1 General structure

644 [RFC3986](#) defines the URI-reference ABNF rule as follows:

```
645 URI-reference = URI / relative-ref
646
647 URI           = scheme ":" hier-part [ "?" query ] [ "#" fragment ]
648
649 relative-ref  = relative-part [ "?" query ] [ "#" fragment ]
650
651 hier-part     = "//" authority path-abempty
652               / path-absolute
653               / path-rootless
654               / path-empty
655
656 relative-part = "//" authority path-abempty
657               / path-absolute
658               / path-noscheme
659               / path-empty
```

660 WBEM resource identifiers shall conform to the following ABNF rule (which conforms to but restricts the
 661 URI-reference ABNF rule defined in [RFC3986](#)):

```
662 WBEM-resource-identifier = [ scheme ":" ] [ "//" authority ]
663                           path-absolute
664                           [ "?" query ]
```

665 Where:

- 666 • `scheme` is defined in [RFC3986](#) and shall in addition conform to the definitions in 7.1.3.
- 667 • `authority` is defined in [RFC3986](#) and shall in addition conform to the definitions in 7.1.4.
- 668 • `path-absolute` is defined in [RFC3986](#) and shall in addition conform to the definitions in 7.1.5.
- 669 • `query` is defined in [RFC3986](#) and shall in addition conform to the definitions in 7.3.

670 7.1.2 Percent-encoding

671 This clause defines how the percent-encoding rules defined in [RFC3986](#) are applied to WBEM resource
672 identifiers.

673 [RFC3986](#) defines percent-encoding in its section 2.1 and defines the set of characters to be percent-
674 encoded in URIs as follows:

- 675 • Any characters outside the character set allowed for URIs (that is, reserved + unreserved) shall
676 be percent-encoded.
- 677 • Any characters in the reserved character set for URIs shall be percent-encoded if data for a URI
678 component would conflict with a reserved character's purpose as a delimiter.
- 679 • Any other characters allowed for URIs should not be percent-encoded.

680 Further, [RFC3986](#) defines the reserved and unreserved character sets for URIs as follows:

```
681 reserved = gen-delims / sub-delims
682
683 gen-delims = ":" / "/"                ; colon (U+003A), slash (U+002F)
684             / "?" / "#"              ; quest.mark (U+003F), hash (U+0023)
685             / "[" / "]"              ; l.bracket (U+005B), r.bracket(U+005D)
686             / "@"                    ; at (U+0040)
687
688 sub-delims = "!" / "$"              ; excl.mark (U+0021), dollar (U+0024)
689             / "&" / "'"              ; ampersand (U+0026), s.quote (U+0027)
690             / "(" / ")"              ; l.paren. (U+0028), r.paren. (U+0029)
691             / "*" / "+"              ; asterisk (U+002A), plus (U+002B)
692             / "," / ";"              ; comma (U+002C), semicolon (U+003B)
693             / "="                    ; equal (U+003D)
694
695 unreserved = ALPHA / DIGIT
696             / "-" / "."              ; hyphen (U+002D), period (U+002E)
697             / "_" / "~"              ; underscore (U+005F), tilde (U+007E)
698
699 ALPHA = %x41-5A / %x61-7A           ; A-Z, a-z
700
701 DIGIT = %x30-39                     ; 0-9
```

702 7.1.2.1 What to percent-encode

703 This clause defines which characters in a WBEM resource identifier can be percent-encoded, consistent
704 with the percent-encoding rules defined in [RFC3986](#).

705 For CIM element names and CIM namespace names used in a WBEM resource identifier, the following
706 percent-encoding rules apply for producers of WBEM resource identifiers:

```
707 npe-chars-for-names = ALPHA / DIGIT / "_"
```

- 708 • Characters that match the ABNF rule `npe-chars-for-names` should not be percent-encoded.
- 709 • Characters that do not match the ABNF rule `npe-chars-for-names` shall be percent-encoded.

710 For CIM key values used in a WBEM resource identifier, the relevance of the reserved characters with
 711 respect to being interpreted as delimiters needs to be defined before the percent-encoding rules defined
 712 in [RFC3986](#) can be applied. In order to prepare for potential future extensions, this document defines that
 713 all characters in the reserved character set for URIs are considered to be (actual or potential future)
 714 delimiters.

715 NOTE The subset of the reserved characters that is currently used as a delimiter of some sort in WBEM resource
 716 identifiers is: ":", "/", "?", "&", ",", ";", "=".

717 Thus, for CIM key values used in a WBEM resource identifier, the following percent-encoding rules apply
 718 for producers of WBEM resource identifiers:

719 `npe-chars-for-keyvalues = ALPHA / DIGIT / "_" / "-" / "." / "~"`

- 720 • Characters that match the ABNF rule `npe-chars-for-keyvalues` should not be percent-
 721 encoded.
- 722 • Characters that do not match the ABNF rule `npe-chars-for-keyvalues` shall be percent-
 723 encoded.

724 Consumers of WBEM resource identifiers shall support any percent-encoding within the WBEM resource
 725 identifier that is permissible according to the rules in this clause.

726 7.1.2.2 How to percent-encode

727 [RFC3986](#) defines percent-encoding on the basis of data octets, but it does not define how characters are
 728 encoded as data octets. Because CIM element names, CIM namespace names, and CIM key values may
 729 contain UCS characters outside of the US-ASCII character set, this document defines the percent-
 730 encoding to be used in WBEM resource identifiers as follows.

731 Any UCS character that is being percent-encoded in WBEM resource identifiers shall be processed by
 732 first normalizing the UCS character using Normalization Form C (defined in [The Unicode Standard, Annex
 733 #15](#)), then encoding it to data octets using UTF-8, and finally percent-encoding the resulting data octets
 734 as defined in section 2.1 of [RFC3986](#). The requirement to use a specific Unicode normalization form and
 735 a specific Unicode encoding (that is, UTF-8) ensures that the resulting string can be compared octet-wise
 736 without having to apply UCS character semantics.

737 For the purpose of representing key values in WBEM resource identifiers, values of CIM data types other
 738 than string shall be represented as string values as defined in [DSP0004](#).

739 7.1.2.3 Examples

740 The following examples use the minimally needed percent-encodings:

- 741 • The CIM namespace name `"root/cimv2"` becomes `"root%2Fcimv2"` in a resource
 742 identifier.
- 743 • The CIM class name `"CIM_LogicalDevice"` remains unchanged in a resource identifier.
- 744 • The (German) CIM method name `"ÄnderungsRate"` becomes `"%C3%84%0AnderungsRate"`
 745 in a resource identifier. (C3 84 0A are the data octets of the UTF-8 encoding of the UCS
 746 character U+00C4, which represents "Ä" in normalized form.)
- 747 • The string typed key value `"a \"brown\" bag\n"` (represented using backslash escape
 748 sequences as defined for string literals in CIM MOF) becomes
 749 `"a%20%22brown%22%20bag%10"` in a resource identifier.
- 750 • The sint8 typed key value `-42` becomes `"-42"` in a resource identifier.

751 **7.1.3 Scheme**

752 WBEM clients, WBEM servers, and WBEM listeners shall adhere to the following additional rules
753 regarding the value of the `scheme` component in the `WBEM-resource-identifier` ABNF rule defined
754 in 7.1.1:

- 755 • A value of "http" shall indicate the use of HTTP as defined in clause 8.21.
- 756 • A value of "https" shall indicate the use of HTTPS as defined in clause 8.21.
- 757 • No other values shall be used.
- 758 • The [`scheme ":"`] part of `WBEM-resource-identifier` may be omitted only if the
759 scheme is known by other means.

760 **7.1.4 Authority**

761 WBEM clients, WBEM servers, and WBEM listeners shall adhere to the following additional rules
762 regarding the value of the `authority` component in the `WBEM-resource-identifier` ABNF rule
763 defined in 7.1.1:

- 764 • The `userinfo` component within `authority` shall not be specified because of security issues
765 with specifying an unencrypted password.
- 766 • The `host` component within `authority` shall be the IP (V4 or V6) address of the WBEM server,
767 or a DNS-resolvable host name for that IP address.
- 768 • If the `port` component within `authority` is not specified, the port shall be assumed as follows:
769 – port number 80 for the scheme "http"
770 – port number 443 for the scheme "https"
- 771 • The [`"/"` `authority`] part of `WBEM-resource-identifier` may be omitted only if the
772 authority is known by other means.

773 **7.1.5 Client knowledge about URI structure**

774 This document describes the structure of WBEM resource identifiers as a requirement for the
775 implementation of the server (WBEM server in its server role, or WBEM listener).

776 A WBEM client should not parse or construct WBEM resource identifiers except for constructing the
777 WBEM resource identifiers of the top-level REST resources (that is, CIM namespaces).

778 A WBEM server in its client role (that is, when interacting with a WBEM listener) gets the WBEM resource
779 identifiers for interacting with the WBEM listener from listener destinations it knows about. As a result, it
780 does not need to construct any WBEM resource identifiers.

781 **7.2 REST resources and their URIs**

782 This clause defines the REST resources used in the CIM-RS protocol and the URIs addressing these
783 resources by means of path components of WBEM resource identifiers. For details on how these path
784 components are used to construct the URIs, see 7.1.

785 This clause defines the `path-absolute` rule by using ABNF incremental alternatives (that is, the `=/`
786 construct), based on the initially empty rule:

```
787 path-absolute = "" ; initially empty
```

788 To avoid ambiguities within a WBEM server that provides multiple access facilities through HTTP, all path
789 components defined in this clause begin with "/cimrs/".

790 The path components defined in this clause not only provide for distinguishing the four types of CIM
791 objects defined in [DSP0004](#) (that is, namespace, class, instance, and qualifier type), but in addition
792 encode some portion of the semantics of the operations defined in [DSP0223](#). As a result, a resource
793 identifier using these path components has sufficient operational semantics to be used with the HTTP
794 methods, as detailed in clause 8.
795

796 7.2.1 Namespace collection resource

797 A *namespace collection resource* represents a set of CIM namespace objects in a WBEM server.

798 The WBEM resource identifier of a namespace collection resource has a defined structure. This allows
799 WBEM clients to use this resource as a starting point (or top-level resource) for interacting with a WBEM
800 server, and to discover further resources from there.

801 A WBEM resource identifier addressing a namespace collection resource shall conform to the WBEM-
802 resource-identifier ABNF rule (see 7.1.1) with the following additional rules:

```
803 path-absolute = NamespaceCollectionPath
804
805 NamespaceCollectionPath = "/cimrs/namespaces"
```

806 Examples:

- 807 • `http://acme.com:5988/cimrs/namespaces`

808 This resource identifier addresses the collection of all CIM namespaces accessible through the
809 WBEM server at port 5988 of host "acme.com", using the "http" scheme.

810 7.2.2 Namespace resource

811 A *namespace resource* represents a CIM namespace object in a WBEM server.

812 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of a namespace
813 resource is implementation-defined (see ANNEX B for examples). The WBEM resource identifiers of
814 namespace resources can be discovered by WBEM clients through links (see 8.2.2).

815 7.2.3 Class collection resource

816 A *class collection resource* represents a set of CIM class objects in a particular CIM namespace in a
817 WBEM server.

818 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of a class
819 collection resource is implementation-defined (see ANNEX B for examples). The WBEM resource
820 identifiers of class collection resources can be discovered by WBEM clients through links (see 8.2.2).

821 7.2.4 Class resource

822 A *class resource* represents a CIM class object in a CIM namespace in a WBEM server.

823 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of a class resource
824 is implementation-defined (see ANNEX B for examples). The WBEM resource identifiers of class
825 resources can be discovered by WBEM clients through links (see 8.2.2).

826 **7.2.5 Class associator collection resource**

827 A *class associator collection resource* represents a set of CIM class objects that are associated with a
828 particular CIM class object in a CIM namespace in a WBEM server. The associated class objects may not
829 all be in the same CIM namespace or WBEM server.

830 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of a class
831 associator collection resource is implementation-defined (see ANNEX B for examples). The WBEM
832 resource identifiers of class associator collection resources can be discovered by WBEM clients through
833 links (see 8.2.2).

834 **7.2.6 Class reference collection resource**

835 A *class reference collection resource* represents the set of CIM association class objects that reference a
836 particular CIM class object in a CIM namespace in a WBEM server. The referencing class objects may
837 not all be in the same CIM namespace or WBEM server.

838 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of a class
839 reference collection resource is implementation-defined (see ANNEX B for examples). The WBEM
840 resource identifiers of class reference collection resources can be discovered by WBEM clients through
841 links (see 8.2.2).

842 **7.2.7 Class method invocation resource**

843 A *class method invocation resource* represents an invocation point for static CIM methods that can be
844 invoked on a CIM class, including the CIM class object in a CIM namespace in a WBEM server, on which
845 the methods can be invoked.

846 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of a class method
847 resource is implementation-defined (see ANNEX B for examples). The WBEM resource identifiers of
848 class method resources can be discovered by WBEM clients through links (see 8.2.2).

849 **7.2.8 Instance collection resource**

850 An *instance collection resource* represents a set of CIM instance objects of a particular CIM class in a
851 CIM namespace in a WBEM server.

852 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of an instance
853 collection resource is implementation-defined (see ANNEX B for examples). The WBEM resource
854 identifiers of instance collection resources can be discovered by WBEM clients through links (see 8.2.2).

855 **7.2.9 Instance resource**

856 An *instance resource* represents a CIM instance object in a CIM namespace in a WBEM server.

857 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of an instance
858 resource is implementation-defined (see ANNEX B for examples). The WBEM resource identifiers of
859 instance resources can be discovered by WBEM clients through links (see 8.2.2).

860 **7.2.10 Instance associator collection resource**

861 An *instance associator collection resource* represents a set of CIM instance objects that are associated
862 with a particular CIM instance object in a CIM namespace in a WBEM server.

863 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of an instance
864 associator collection resource is implementation-defined (see ANNEX B for examples). The WBEM
865 resource identifiers of instance associator collection resources can be discovered by WBEM clients
866 through links (see 8.2.2).

867 **7.2.11 Instance reference collection resource**

868 An *instance reference collection resource* represents the set of CIM association instance objects that
869 reference a particular CIM instance object in a CIM namespace in a WBEM server.

870 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of an instance
871 reference collection resource is implementation-defined (see ANNEX B for examples). The WBEM
872 resource identifiers of instance reference collection resources can be discovered by WBEM clients
873 through links (see 8.2.2).

874 **7.2.12 Instance method invocation resource**

875 An *instance method invocation resource* represents an invocation point for (static or non-static) CIM
876 methods that can be invoked on a CIM instance, including the CIM instance object in a CIM namespace
877 in a WBEM server on which the methods can be invoked.

878 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of an instance
879 method resource is implementation-defined (see ANNEX B for examples). The WBEM resource identifiers
880 of instance method resources can be discovered by WBEM clients through links (see 8.2.2).

881 **7.2.13 Qualifier type collection resource**

882 A *qualifier type collection resource* represents the set of all CIM qualifier type objects in a particular CIM
883 namespace in a WBEM server.

884 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of a qualifier type
885 collection resource is implementation-defined (see ANNEX B for examples). The WBEM resource
886 identifiers of qualifier type collection resources can be discovered by WBEM clients through links (see
887 8.2.2).

888 **7.2.14 Qualifier type resource**

889 A *qualifier type resource* represents a CIM qualifier type object in a CIM namespace in a WBEM server.

890 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of a qualifier type
891 resource is implementation-defined (see ANNEX B for examples). The WBEM resource identifiers of
892 qualifier type resources can be discovered by WBEM clients through links (see 8.2.2).

893 **7.2.15 Instance query resource**

894 An *instance query resource* represents a target for executing queries on CIM instances in a CIM
895 namespace in a WBEM server.

896 Beyond the requirements defined in 7.1, the structure of the WBEM resource identifier of an instance
 897 query resource is implementation-defined (see ANNEX B for examples). The WBEM resource identifiers
 898 of instance query resources can be discovered by WBEM clients through links (see 8.2.2).

899 7.3 Query parameters in URIs

900 This clause defines the query component of WBEM resource identifiers, and applies in addition to the
 901 definition in [RFC3986](#), section 3.4.

902 The representation of the query component is defined by the following ABNF rules:

```
903 query = query-parameter *( "&" query-parameter )
```

904 Where:

- 905 • `query-parameter` is a query parameter defined in the subclasses of 7.37.3.

906 Example:

- 907 • `/cimrs/namespaces/myns/classes?spc=CIM_System`

908 This resource identifier addresses the list of classes in namespace "myns" whose superclass is
 909 CIM_System. (For details on the `spc` query parameter, see 7.3.17.)

910 Query parameters of URIs are case sensitive, as defined in [RFC3986](#), section 6.2.2.1. As a result, two
 911 URIs that differ only in the lexical case of query parameters do not match. The query parameters defined
 912 in the subclasses of 7.3 define in some cases that the values of query parameters are to be treated case
 913 insensitively. In such cases, two URIs that differ only in the lexical case of query parameters address the
 914 same resource, even though the URIs do not match according to the rules defined in [RFC3986](#). It is
 915 recommended that producers of URIs use the lexical case for such element names as specified in the
 916 underlying CIM schema, in order to optimize caching based on URIs. For example, if a class is named
 917 "CIM_LogicalDevice", its use in the `scn` query parameter would be "`scn=CIM_LogicalDevice`".

918 Query parameters (identified by their name) shall not be repeated in WBEM resource identifiers. If query
 919 parameters are repeated in a WBEM resource identifier that is used as the target of an operation, the
 920 CIM-RS consumer shall reject that operation.

921 NOTE: [RFC3986](#) does not detail how the `query` ABNF rule is broken into query parameters, and thus does not
 922 address the topic of query parameter repetition.

923 This clause defines the `query-parameter` rule by using ABNF incremental alternatives (that is, the `=/`
 924 construct), based on the initially empty rule:

```
925 query-parameter = "" ; initially empty
```

926 7.3.1 acn (associated class name)

927 The `acn` query parameter acts as a restricting filter on the set of instances or classes returned by
 928 association traversal operations, as defined in the description of the `AssociatedClassName` generic
 929 operation parameter in [DSP0223](#).

930 The representation of this query parameter is defined by the following ABNF:

```
931 query-parameter =/ AssociatedClassName-queryparm
```

932

```
933 AssociatedClassName-queryparm = [ "acn=" AssociatedClassName ]
```

934 Where:

- 935 • `AssociatedClassName` is the name of the associated class on the far end of the association as
936 seen from the source class or instance (including its schema prefix).

937 An `acn` query parameter that is not specified in the resource identifier shall correspond to an
938 `AssociatedClassName` generic operation parameter value of Null.

939 An `acn` query parameter that is specified in the resource identifier with a value shall correspond to the
940 equivalent `AssociatedClassName` generic operation parameter value.

941 NOTE: While query parameters of URIs are case sensitive, as defined in [RFC3986](#) (section 6.2.2.1), class names are
942 to be interpreted case insensitively, as defined in [DSP0004](#).

943 Examples:

944 " " (not specified) – Null

945 "`acn=CIM_Endpoint1`" – associated class name of "`CIM_Endpoint1`"

946 7.3.2 `arn` (associated role name)

947 The `arn` query parameter acts as a restricting filter on the set of instances or classes returned by
948 association traversal operations, as defined in the description of the *AssociatedRoleName* generic
949 operation parameter in [DSP0223](#).

950 The representation of this query parameter is defined by the following ABNF:

```
951 query-parameter =/ AssociatedRoleName-queryparm
952
953 AssociatedRoleName-queryparm = [ "arn=" AssociatedRoleName ]
```

954 Where:

- 955 • `AssociatedRoleName` is the name of the role (that is, reference) of the associated class or
956 instance.

957 An `arn` query parameter that is not specified in the resource identifier shall correspond to an
958 `AssociatedRoleName` generic operation parameter value of Null.

959 An `arn` query parameter that is specified in the resource identifier with a value shall correspond to the
960 equivalent `AssociatedRoleName` generic operation parameter value.

961 NOTE: While query parameters of URIs are case sensitive, as defined in [RFC3986](#) (section 6.2.2.1), role (that is,
962 reference) names are to be interpreted case insensitively, as defined in [DSP0004](#).

963 Examples:

964 " " (not specified) – Null

965 "`arn=Child`" – associated role (that is, reference) name "`Child`"

966 7.3.3 `c` (class)

967 The `c` query parameter acts as a restricting filter on the set of classes returned by operations to the one
968 that has the class name specified in the `c` query parameter. It has no directly corresponding generic
969 operation parameter in [DSP0223](#).

970 The representation of this query parameter is defined by the following ABNF:

```
971 query-parameter =/ Class-queryparm
```

972
 973 `Class-queryparm = ["c=" ClassName]`

974 Where:

- 975 • `ClassName` is the name of the class (including its schema prefix).

976 A `c` query parameter that is not specified in the resource identifier shall not restrict the returned classes.

977 A `c` query parameter that is specified in the resource identifier shall restrict the returned classes to the
 978 one that has the class name specified in the `c` query parameter.

979 NOTE: While query parameters of URIs are case sensitive, as defined in [RFC3986](#) (section 6.2.2.1), class names are
 980 to be interpreted case insensitively, as defined in [DSP0004](#).

981 Examples:

982 `" "` (not specified) – no restriction

983 `"c=CIM_LogicalElement"` – restrict returned classes to "CIM_LogicalElement"

984 7.3.4 **dd (delete dependents)**

985 The `dd` query parameter controls whether dependent classes shall be deleted, as defined in the
 986 description of the *DeleteDependents* generic operation parameter in [DSP0223](#).

987 The representation of this query parameter is defined by the following ABNF:

```
988 query-parameter =/ DeleteDependents-queryparm
989
990 DeleteDependents-queryparm = [ "dd" [ "=" ( "true" / "false" ) ] ]
```

991 A `dd` query parameter that is not specified in the resource identifier shall correspond to a
 992 *DeleteDependents* generic operation parameter value of False.

993 A `dd` query parameter that is specified in the resource identifier without a value shall correspond to a
 994 *DeleteDependents* generic operation parameter value of True.

995 A `dd` query parameter that is specified in the resource identifier with a value shall correspond to the
 996 equivalent *DeleteDependents* generic operation parameter value.

997 The value of the `dd` query parameter, if specified, shall be either "true" or "false".

998 Examples:

999 `" "` (not specified) – False

1000 `"dd"` – True

1001 `"dd=false"` – False

1002 `"dd=true"` – True

1003 7.3.5 **esbp (exclude subclass properties)**

1004 The *esbp* query parameter acts as a restricting filter on the set of properties that is included in any
 1005 returned instances or classes, as defined in the description of the *ExcludeSubclassProperties* generic
 1006 operation parameter in [DSP0223](#).

1007 The representation of this query parameter is defined by the following ABNF:

```
1008 query-parameter =/ ExcludeSubclassProperties-queryparm
1009
1010 ExcludeSubclassProperties-queryparm = [ "esbp" [ "=" ( "true"/"false" ) ] ]
```

1011 An esbp query parameter that is not specified in the resource identifier shall correspond to an
1012 ExcludeSubclassProperties generic operation parameter value of False (causing subclass properties not
1013 to be excluded).

1014 An esbp query parameter that is specified in the resource identifier without a value shall correspond to an
1015 ExcludeSubclassProperties generic operation parameter value of True (causing subclass properties to be
1016 excluded).

1017 An esbp query parameter that is specified in the resource identifier with a value shall correspond to the
1018 equivalent ExcludeSubclassProperties generic operation parameter value.

1019 The value of the esbp query parameter, if specified, shall be either "true" or "false".

1020 Examples:

1021 " " (not specified) – False

1022 "esbp" – True

1023 "esbp=false" – False

1024 "esbp=true" – True

1025 7.3.6 fql (filter query language)

1026 The *fql* query parameter specifies the query language used for the value of the *fqs* (filter query string)
1027 query parameter, as defined in the description of the *FilterQueryLanguage* generic operation parameter in
1028 [DSP0223](#).

1029 The representation of this query parameter is defined by the following ABNF:

```
1030 query-parameter =/ FilterQueryLanguage-queryparm
1031
1032 FilterQueryLanguage-queryparm = [ "fql=" FilterQueryLanguage ];
```

1033 An fql query parameter that is not specified in the resource identifier shall correspond to a
1034 FilterQueryLanguage generic operation parameter value of NULL.

1035 An fql query parameter that is specified in the resource identifier with a value shall correspond to the
1036 equivalent FilterQueryLanguage generic operation parameter value.

1037 Examples:

1038 " " (not specified) – NULL

1039 "fql=DMTF%3ACQL" – query language "DMTF:CQL"

1040 7.3.7 fqs (filter query string)

1041 The *fqs* query parameter specifies the query string of a filter query for operations that return instance
1042 collections, in the query language indicated by the *fql* query parameter.

1043 The filter query acts as an additional restricting filter on the set of instances returned, as defined in the
1044 description of the *FilterQueryString* generic operation parameter in [DSP0223](#).

1045 The representation of this query parameter is defined by the following ABNF:

```
1046 query-parameter =/ FilterQueryString-queryparm
1047
1048 FilterQueryString-queryparm = [ "fqs=" QueryString ]
```

1049 An fqs query parameter that is not specified in the resource identifier shall correspond to a
1050 *FilterQueryString* generic operation parameter value of NULL.

1051 An fqs query parameter that is specified in the resource identifier with a value shall correspond to the
1052 equivalent *FilterQueryString* generic operation parameter value. [DSP0223](#) defines restrictions on the filter
1053 query.

1054 Examples:

```
1055     " " (not specified) – NULL
1056     "fqs=SELECT%20%2A%20FROM%20CIM_StorageExtent%20WHERE%20Name%3D'abc' "
1057     – CQL query "SELECT * FROM CIM_StorageExtent WHERE Name='abc'"
```

1058 **7.3.8 ico (include class origin)**

1059 The *ico* query parameter controls whether class origin information is returned for any elements in returned
1060 instances or classes, as defined in the description of the *IncludeClassOrigin* generic operation parameter
1061 in [DSP0223](#).

1062 The representation of this query parameter is defined by the following ABNF:

```
1063 query-parameter =/ IncludeClassOrigin-queryparm
1064
1065 IncludeClassOrigin-queryparm = [ "ico" [ "=" ( "true" / "false" ) ] ]
```

1066 An *ico* query parameter that is not specified in the resource identifier shall correspond to an
1067 *IncludeClassOrigin* generic operation parameter value of False.

1068 An *ico* query parameter that is specified in the resource identifier without a value shall correspond to an
1069 *IncludeClassOrigin* generic operation parameter value of True.

1070 An *ico* query parameter that is specified in the resource identifier with a value shall correspond to the
1071 equivalent *IncludeClassOrigin* generic operation parameter value.

1072 The value of the *ico* query parameter, if specified, shall be either "true" or "false".

1073 Examples:

```
1074     " " (not specified) – False
1075     "ico" – True
1076     "ico=false" – False
1077     "ico=true" – True
```

1078 7.3.9 iie (include inherited elements)

1079 The *iie* query parameter controls whether elements inherited from superclasses are included in any
 1080 returned classes, as defined in the description of the *IncludeInheritedElements* generic operation
 1081 parameter in [DSP0223](#).

1082 The representation of this query parameter is defined by the following ABNF:

```
1083 query-parameter =/ IncludeInheritedElements-queryparm
1084
1085 IncludeInheritedElements-queryparm = [ "iie" [ "=" ("true"/"false") ] ]
```

1086 An *iie* query parameter that is not specified in the resource identifier shall correspond to an
 1087 *IncludeInheritedElements* generic operation parameter value of False (causing inherited elements not to
 1088 be included).

1089 An *iie* query parameter that is specified in the resource identifier without a value shall correspond to an
 1090 *IncludeInheritedElements* generic operation parameter value of True (causing inherited elements to be
 1091 included).

1092 An *iie* query parameter that is specified in the resource identifier with a value shall correspond to the
 1093 equivalent *IncludeInheritedElements* generic operation parameter value.

1094 The value of the *iie* query parameter, if specified, shall be either "true" or "false".

1095 Examples:

1096 " " (not specified) – False

1097 "iie" – True

1098 "iie=false" – False

1099 "iie=true" – True

1100 7.3.10 ip (included properties)

1101 The *ip* query parameter acts as a restricting filter on the set of properties that is included in any returned
 1102 instances or classes, as defined in the description of the *IncludedProperties* generic operation parameter
 1103 in [DSP0223](#).

1104 Its representation in the query component is defined by the following ABNF:

```
1105 query-parameter =/ IncludedProperties-queryparm
1106
1107 IncludedProperties-queryparm = [ "ip=" PropertyNameList ]
```

1108 Where:

- 1109 • *PropertyNameList* is a comma-separated list of property names. The list may be empty.

1110 An *ip* query parameter that is not specified in the resource identifier shall correspond to an
 1111 *IncludedProperties* generic operation parameter value of Null (causing no restriction on properties).

1112 An *ip* query parameter that is specified in the resource identifier without a value shall correspond to an
 1113 *IncludedProperties* generic operation parameter value of the empty array (causing all properties to be
 1114 excluded).

1115 An ip query parameter that is specified in the resource identifier with a value shall correspond to the
 1116 equivalent IncludedProperties generic operation parameter value, where each property name in the list in
 1117 the query parameter value shall correspond to an array entry in the generic operation parameter value
 1118 (causing properties not specified to be excluded).

1119 NOTE: While query parameters of URIs are case sensitive, as defined in [RFC3986](#) (section 6.2.2.1), property names
 1120 are to be interpreted case insensitively, as defined in [DSP0004](#).

1121 Examples:

1122 " " (not specified) – Null

1123 "ip=" – empty list

1124 "ip=Prop1, Prop2" – list of Prop1, Prop2

1125 7.3.11 iq (include qualifiers)

1126 The *iq* query parameter controls whether qualifier information is returned for any elements in returned
 1127 instances or classes, as defined in the description of the *IncludeQualifiers* generic operation parameter in
 1128 [DSP0223](#).

1129 The representation of this query parameter is defined by the following ABNF:

```
1130 query-parameter =/ IncludeQualifiers-queryparm
```

1131

```
1132 IncludeQualifiers-queryparm = [ "iq" [ "=" ( "true" / "false" ) ] ]
```

1133 An iq query parameter that is not specified in the resource identifier shall correspond to an
 1134 IncludeQualifiers generic operation parameter value of False.

1135 An iq query parameter that is specified in the resource identifier without a value shall correspond to an
 1136 IncludeQualifiers generic operation parameter value of True.

1137 An iq query parameter that is specified in the resource identifier with a value shall correspond to the
 1138 equivalent IncludeQualifiers generic operation parameter value.

1139 The value of the iq query parameter, if specified, shall be either "true" or "false".

1140 Examples:

1141 " " (not specified) – False

1142 "iq" – True

1143 "iq=false" – False

1144 "iq=true" – True

1145 7.3.12 isbc (include subclasses)

1146 The *isbc* query parameter controls whether the entire tree of subclasses is included in any returned
 1147 classes, as defined in the description of the *IncludeSubclasses* generic operation parameter in [DSP0223](#).

1148 The representation of this query parameter is defined by the following ABNF:

```
1149 query-parameter =/ IncludeSubclasses-queryparm
```

1150

1151 `IncludeSubclasses-queryparm = ["isbc" ["=" ("true" / "false")]]`

1152 An isbc query parameter that is not specified in the resource identifier shall correspond to an
1153 IncludeSubclasses generic operation parameter value of False.

1154 An isbc query parameter that is specified in the resource identifier without a value shall correspond to an
1155 IncludeSubclasses generic operation parameter value of True.

1156 An isbc query parameter that is specified in the resource identifier with a value shall correspond to the
1157 equivalent IncludeSubclasses generic operation parameter value.

1158 The value of the isbc query parameter, if specified, shall be either "true" or "false".

1159 Examples:

1160 " " (not specified) – False

1161 "isbc" – True

1162 "isbc=false" – False

1163 "isbc=true" – True

1164 7.3.13 n (namespace)

1165 The *n* query parameter acts as a restricting filter on the set of namespaces returned by operations to the
1166 one that has the namespace name specified in the *n* query parameter. It has no directly corresponding
1167 generic operation parameter in [DSP0223](#).

1168 The representation of this query parameter is defined by the following ABNF:

```
1169 query-parameter =/ Namespace-queryparm
1170
1171 Namespace-queryparm = [ "n=" NamespaceName ]
```

1172 Where:

- 1173 • NamespaceName is the name of the namespace (e.g. "root%2Fcimv2").

1174 An *n* query parameter that is not specified in the resource identifier shall not restrict the returned
1175 namespaces.

1176 An *n* query parameter that is specified in the resource identifier shall restrict the returned namespaces to
1177 the one that has the namespace name specified in the *n* query parameter.

1178 NOTE: While query parameters of URIs are case sensitive, as defined in [RFC3986](#) (section 6.2.2.1), namespace
1179 names are to be interpreted case insensitively, as defined in [DSP0004](#).

1180 Examples:

1181 " " (not specified) – no restriction

1182 "n=root%2F/cimv2" – restrict returned namespaces to "root/cimv2"

1183 7.3.14 pr (paged retrieval)

1184 The *pr* query parameter controls whether paged retrieval of resources is requested (see 8.3.2).

1185 The representation of this query parameter is defined by the following ABNF:

1186 `query-parameter =/ PagedRetrieval-queryparm`

1187

1188 `PagedRetrieval-queryparm = ["pr" ["=" enumerationcontext]]`

1189 A pr query parameter that is not specified in the resource identifier means that no paged retrieval is
1190 requested. In other words, the entire resource is requested to be retrieved.

1191 A pr query parameter that is specified in the resource identifier without a value means that paged retrieval
1192 of the first subset of the resource is requested. In other words, this opens an enumeration.

1193 A pr query parameter that is specified in the resource identifier with a value means that paged retrieval is
1194 requested of the subset of the resource that is identified by the value. The query parameter value shall be
1195 the base64url-encoded enumeration context value without any "=" padding characters at the end. The
1196 base64url encoding is defined in [RFC4648](#).

1197 Examples:

1198 " " (not specified) – No paged retrieval is requested

1199 "pr" – Paged retrieval of the first subset of the resource is requested

1200 "pr=YWJjZGVm" – Paged retrieval of the resource subset identified by enumeration context value
1201 "abcdef" is requested

1202 7.3.15 q (qualifier)

1203 The q query parameter acts as a restricting filter on the set of qualifier types returned by operations to the
1204 one that has the qualifier name specified in the q query parameter. It has no directly corresponding
1205 generic operation parameter in [DSP0223](#).

1206 The representation of this query parameter is defined by the following ABNF:

1207 `query-parameter =/ Qualifier-queryparm`

1208

1209 `Qualifier-queryparm = ["q=" QualifierName]`

1210 Where:

- 1211 • QualifierName is the name of the qualifier type (e.g. "Alias").

1212 A q query parameter that is not specified in the resource identifier shall not restrict the returned qualifier
1213 types.

1214 A q query parameter that is specified in the resource identifier shall restrict the returned qualifier types to
1215 the one that has the qualifier name specified in the n query parameter.

1216 NOTE: While query parameters of URIs are case sensitive, as defined in [RFC3986](#) (section 6.2.2.1), qualifier names
1217 are to be interpreted case insensitively, as defined in [DSP0004](#).

1218 Examples:

1219 " " (not specified) – no restriction

1220 "q=Alias" – restrict returned qualifier types to "Alias"

1221 7.3.16 rcn (referencing class name)

1222 The *rcn* query parameter acts as a restricting filter on the set of instances or classes returned by
 1223 association traversal operations, as defined in the description of the *AssociationClassName* generic
 1224 operation parameter in [DSP0223](#).

1225 The representation of this query parameter is defined by the following ABNF:

```
1226 query-parameter =/ ReferencingClassName-queryparm
1227
1228 ReferencingClassName-queryparm = [ "rcn=" ReferencingClassName ]
```

1229 Where:

- 1230 • *ReferencingClassName* is the name of the association class referencing the source instance
 1231 or class (including its schema prefix).

1232 An *rcn* query parameter that is not specified in the resource identifier shall correspond to an
 1233 *AssociationClassName* generic operation parameter value of Null.

1234 An *rcn* query parameter that is specified in the resource identifier with a value shall correspond to the
 1235 equivalent *AssociationClassName* generic operation parameter value.

1236 NOTE: While query parameters of URIs are case sensitive, as defined in [RFC3986](#) (section 6.2.2.1), class names are
 1237 to be interpreted case insensitively, as defined in [DSP0004](#).

1238 Examples:

1239 " " (not specified) – Null

1240 "rcn=CIM_Assoc1" – referencing class name of "CIM_Assoc1"

1241 7.3.17 spc (superclass)

1242 The *spc* query parameter acts as a restricting filter on the set of classes returned by operations to those
 1243 that have the superclass specified in the *spc* query parameter. It has no directly corresponding generic
 1244 operation parameter in [DSP0223](#).

1245 The representation of this query parameter is defined by the following ABNF:

```
1246 query-parameter =/ Superclass-queryparm
1247
1248 Superclass-queryparm = [ "spc=" SuperClassName ]
```

1249 Where:

- 1250 • *SuperClassName* is the name of the superclass (including its schema prefix).

1251 An *spc* query parameter that is not specified in the resource identifier shall not restrict the returned
 1252 classes.

1253 An *spc* query parameter that is specified in the resource identifier shall restrict the returned classes to
 1254 those that have the specified superclass. In other words, the returned classes are restricted to subclasses
 1255 of the specified class.

1256 NOTE: While query parameters of URIs are case sensitive, as defined in [RFC3986](#) (section 6.2.2.1), class names are
 1257 to be interpreted case insensitively, as defined in [DSP0004](#).

1258 Examples:

1259 " " (not specified) – no restriction

1260 "spc=CIM_LogicalElement" – restrict returned classes to subclasses of "CIM_LogicalElement"

1261 **7.3.18 srn (source role name)**

1262 The *srn* query parameter acts as a restricting filter on the set of instances or classes returned by
 1263 association traversal operations, as defined in the description of the *SourceRoleName* or *RoleName*
 1264 generic operation parameter in [DSP0223](#), depending on the actual operation.

1265 The representation of this query parameter is defined by the following ABNF:

```
1266 query-parameter =/ SourceRoleName-queryparm
1267
1268 SourceRoleName-queryparm = [ "srn=" SourceRoleName ]
```

1269 Where:

- 1270 • *SourceRoleName* is the name of the role (that is, reference) of the source class or instance.

1271 An *srn* query parameter that is not specified in the resource identifier shall correspond to a
 1272 *SourceRoleName* or *RoleName* generic operation parameter value of Null.

1273 An *srn* query parameter that is specified in the resource identifier with a value shall correspond to the
 1274 equivalent *SourceRoleName* or *RoleName* generic operation parameter value.

1275 NOTE: While query parameters of URIs are case sensitive, as defined in [RFC3986](#) (section 6.2.2.1), role (that is,
 1276 reference) names are to be interpreted case insensitively, as defined in [DSP0004](#).

1277 Examples:

1278 " " (not specified) – Null

1279 "srn=Parent" – source role (that is, reference) name "Parent"

1280 **8 Operations**

1281 This clause defines the operations of the CIM-RS protocol.

1282 **8.1 Overview on REST resources and HTTP methods**

1283 The operations of the CIM-RS protocol are defined as HTTP methods on certain REST resources, as
 1284 listed in Table 1.

1285 **Table 1 – REST resources and HTTP methods**

Target REST Resource	HTTP Methods
WBEM server	OPTIONS, see 8.5.1
Namespace collection resource, see 7.2.1	GET, see 8.6.1
Namespace resource, see 7.2.2	GET, see 8.7.1
Class collection resource, see 7.2.3	GET, see 8.8.1
	POST, see 8.8.2

Target REST Resource	HTTP Methods
Class resource, see 7.2.4	GET, see 8.9.1
	DELETE, see 8.9.2
	PUT, see 8.9.3
Class associator collection resource, see 7.2.5	GET, see 8.10.1
Class reference collection resource, see 7.2.6	GET, see 8.11.1
Class method invocation resource, see 7.2.7	POST, see 8.12.1
Instance collection resource, see 7.2.8	GET, see 8.13.1
	POST, see 8.13.2
Instance resource, see 7.2.9	GET, see 8.14.1
	DELETE, see 8.14.2
	PUT, see 8.14.3
	PATCH, see 8.14.4
Instance associator collection resource, see 7.2.10	GET, see 8.15.1
Instance reference collection resource, see 7.2.11	GET, see 8.16.1
Instance method invocation resource, see 7.2.12	POST, see 8.17.1
Qualifier type collection resource, see 7.2.13	GET, see 8.18.1
	POST, see 8.18.2
Qualifier type resource, see 7.2.14	GET, see 8.19.1
	DELETE, see 8.19.2
	PUT, see 8.19.3
Instance query resource, see 7.2.15	POST, see 8.20.1

1286

1287 **8.2 Common behaviors for all operations**

1288 **8.2.1 Content negotiation**

1289 WBEM clients, WBEM servers, and WBEM listeners shall support server-driven content negotiation as
 1290 defined in [RFC2616](#), based on the Accept request header field (defined in [RFC2616](#) and in 9.4.1), and
 1291 the Content-Type response header field (defined in [RFC2616](#) and in 9.4.2).

1292 Requirements for the media types used in these header fields are defined in 10.1.

1293 The HTTP OPTIONS method can be used to retrieve the set of CIM-RS payload representations
 1294 supported by a WBEM server or WBEM listener, as described in 8.5.1.

1295 **8.2.2 Links**

1296 Some payload elements in an HTTP entity body include WBEM resource identifiers to the resource
 1297 represented by the payload element, or to related resources or resource collections. Such WBEM
 1298 resource identifiers in payload elements are called *links*.

1299 Links have a name. The name of a link is case sensitive. Link names, their semantic, and the
 1300 requirements for including certain named links in the payload element are defined in this document.

1301 Links have a target. The target of a link is a resource or resource collection that can be operated on using
 1302 CIM-RS operations.

1303 The descriptions of payload elements in 8.4 define requirements for the inclusion of links.

1304 If the paged retrieval feature is implemented, additional links are included in some payload elements, as
 1305 shown in Table 4. shows an overview of links included in payload elements, and the resources targeted
 1306 by these links.

1307 If the paged retrieval feature is implemented, additional links are included in some payload elements, as
 1308 shown in Table 4.

1309 **Table 2 – Links included in payload elements**

Payload element	Links included	Resource targeted by link
NamespaceCollection, see 8.4.1	"self"	the represented namespace collection itself
Namespace, see 8.4.2	"self"	the represented namespace itself
	"classes"	class collection of all classes in the represented namespace
	"qualifiers"	qualifier type collection of all qualifier types in the represented namespace
	"instancequery"	instance query resource of the represented namespace
ClassCollection, see 8.4.3	"self"	the represented class collection itself
	"namespace"	namespace of the represented class collection
Class, see 8.4.4	"self"	the represented class itself
	"namespace"	namespace of the represented class
	"instances"	instance collection of all instances of the represented class
	"methodinvocation"	class method invocation resource for all methods that can be invoked on the represented class
	"associators"	class collection of all classes associated to the represented class
	"references"	class collection of all classes referencing the represented class
InstanceCollection, see 8.4.5	"self"	the represented instance collection itself
	"class"	common superclass of the creation classes of the instances in the represented instance collection
Instance, see 8.4.6	"self"	the represented instance itself
	"class"	creation class of the represented instance

Payload element	Links included	Resource targeted by link
	"methodinvocation"	instance method invocation resource for all methods that can be invoked on the represented instance
	"associators"	instance collection of all instances associated to the represented instance
	"references"	instance collection of all instances referencing the represented instance
QualifierTypeCollection, see 8.4.7	"self"	the represented qualifier type collection itself
	"namespace"	namespace of the represented qualifier type collection
QualifierType, see 8.4.8	"self"	the represented qualifier type itself
	"namespace"	namespace of the represented qualifier type
MethodInvocationRequest, see 8.4.9	none	N/A
MethodInvocationResponse, see 8.4.10	none	N/A
InstanceModificationRequest, see 8.4.11	none	N/A
InstanceQueryRequest, see 8.4.12	none	N/A

1310

1311 **EXPERIMENTAL**

1312 **8.2.3 Verifying the basis of resource modifications**

1313 The PUT instance operation (see 8.14.3) supplies a modified instance as input. The CIM-RS protocol
 1314 provides for a means to verify for a WBEM server whether the current state of the resource is still the
 1315 same as when the WBEM client retrieved the resource as a basis for providing the modified resource.

1316 This is achieved by using the value of the CIM Generation property (defined in CIM_ManagedElement) as
 1317 an entity tag with the Etag and If-Match HTTP header fields, as described in 9.4.3 and 9.4.4.

1318 This ability is experimental, because the Generation property is defined as experimental in the current
 1319 CIM Schema.

1320 This ability is part of the optional entity tagging feature (see 8.3.1).

1321 **EXPERIMENTAL**

1322 **8.2.4 Caching of responses**

1323 Caching of responses from WBEM servers and WBEM listeners is described in [RFC2616](#). This document
 1324 does not define any additional constraints or restrictions on caching.

1325 Implementing the entity tagging feature (see 8.3.1) improves cache control.

1326 **8.2.5 Success and failure**

1327 Operations performed within the CIM-RS protocol either succeed or fail. There is no concept of "partial
1328 success".

1329 If an operation succeeds, it returns its output data to the operation requester and does not include any
1330 error messages.

1331 If it fails, it returns one or more error messages and no output data.

1332 For example, if an instance enumeration operation were able to return some, but not all, instances
1333 successfully, then the operation fails without returning any instances.

1334 **8.2.6 Error messages**

1335 HTTP status codes are used to convey errors. The definition of HTTP status codes defined in [RFC2616](#) is
1336 the basis for each operation, and the operation description may specify additional constraints on the use
1337 of HTTP status codes.

1338 NOTE: The specification of extended error information (for example, using an ordered list of embedded CIM_Error
1339 instances) is not described in this version of the document.

1340

1341 **8.2.7 Consistency model**

1342 The operations of the CIM-RS protocol shall conform to the consistency model defined in [DSP0223](#).

1343 **8.2.8 Common operation parameters for all operations**

1344 This clause defines commonly used operation parameters for the operations. The description of the
1345 individual operations references these parameters as appropriate. Not every operation uses every one of
1346 these common parameters.

1347 **8.3 Optional behaviors for CIM-RS protocol**

1348 This clause defines optional behaviors for the implementation of the CIM-RS protocol.

1349 **8.3.1 Entity tagging feature**

1350 Implementation of the entity tagging feature in WBEM servers and WBEM clients provides for verifying
1351 the basis of resource modifications and thus for improved consistency control in instance modifications,
1352 as described in 8.2.3, and for improved cache control, as described in 8.2.4.

1353 Implementation of the entity tagging feature is optional for WBEM clients and WBEM servers,
1354 independently.

1355 Implementation of the entity tagging feature in a WBEM server is indicated through the CIMRS-Entity-
1356 Tagging-Feature header field (see 9.4.6).

1357 **8.3.2 Paged retrieval feature**

1358 Implementation of the paged retrieval feature in WBEM servers and WBEM clients supports *pulled*
1359 *enumerations* as defined in [DSP0223](#), that is, the retrieval of a result set through multiple HTTP GET
1360 methods, each of which retrieves a subset of the result set.

1361 The concept for paged retrieval is consistent with the concept for paged feeds defined in [RFC5005](#). The
 1362 consistency of the result set is defined in [DSP0223](#) (again, consistent with [RFC5005](#)).

1363 Implementation of the paged retrieval feature is optional for WBEM clients and WBEM servers,
 1364 independently.

1365 Implementation of the paged retrieval feature in a WBEM server is indicated through the CIMRS-Paged-
 1366 Retrieval-Feature header field (see 9.4.7).

1367 CIM-RS supports both modes of maintaining the enumeration state that are defined by [DSP0223](#). The
 1368 WBEM server decides about the mode that is used, and that decision is transparent to the WBEM client:

- 1369 • If the WBEM server maintains the enumeration state, it returns to the WBEM client a reference to
 1370 that state as the value of an enumeration context.
- 1371 • If the WBEM server does not maintain the enumeration state (that is, it is stateless with respect to
 1372 the enumeration state), it returns to the WBEM client the entire enumeration state by value, as
 1373 the value of an enumeration context. The enumeration context value may change as the
 1374 enumeration proceeds.

1375 WBEM servers implementing paged retrieval should implement the mode where they do not maintain the
 1376 enumeration state.

1377 In both modes, the enumeration context for paged retrieval of a resource is represented in the WBEM
 1378 resource identifier of the resource, through the *pr* query parameter (see 7.3.13).

1379 The enumeration context value is passed to the WBEM server by means of the *pr* query parameter in the
 1380 WBEM resource identifier referencing the resource to be retrieved. A "next" link in the result set of a
 1381 paged retrieval is used to pass the new enumeration context value back to the WBEM client. In order to
 1382 retrieve the next paged retrieval subset, the WBEM client uses the WBEM resource identifier specified in
 1383 the "next" link as the target of the HTTP GET method.

1384 In both modes, a WBEM server shall choose the value of the enumeration context such that it is different
 1385 for each distinct paged retrieval subset within an enumeration. This allows a WBEM server to distinguish
 1386 between a request to retrieve the next subset, and a request to repeat the retrieval of the last subset. A
 1387 WBEM server shall not reject requests to retrieve a paged retrieval subset that had already been
 1388 retrieved previously, as identified by the enumeration context value. This rule is motivated by the need to
 1389 be able to repeat requests and by the need for the HTTP GET method to be idempotent.

1390 This version of this document limits paged retrieval to CIM instances, consistent with the current version
 1391 of [DSP0223](#). The following CIM-RS operations support paged retrieval of their result:

1392 **Table 3 – Operations supporting paged retrieval**

Operation	Retrieved resource	Description
GET instance collection	instance collection	see 8.13.1
GET instance associator collection	instance collection	see 8.15.1
GET instance reference collection	instance collection	see 8.16.1

1393

1394 If paged retrieval is implemented, it shall be supported for all operations listed in Table 3.

1395 The actual use of paged retrieval is negotiated between WBEM client and WBEM server. A WBEM client
 1396 initiates paged retrieval of a resource by specifying the *pr* query parameter without a value on a HTTP
 1397 GET method. A WBEM server that does not support paged retrieval shall ignore the presence of the *pr*
 1398 query parameter and shall proceed as if no paged retrieval had been requested. A WBEM server that

1399 supports paged retrieval shall return the paged retrieval subset identified by the pr query parameter and
 1400 shall include the links described in Table 4. The pr query parameter value is the enumeration context
 1401 value in an encoded form, for details see 7.3.13.

1402 **Table 4 – Additional links for paged retrieval**

Link name	Absolute / Relative	Requirement	Targeted resource
first	Absolute or relative to self	Conditional	First paged retrieval subset of result set. Condition: The WBEM server supports paged retrieval.
next	Absolute or relative to self	Conditional	Next paged retrieval subset of result set. Condition: The WBEM server supports paged retrieval and the enumeration is not yet exhausted.

1403
 1404 NOTE: In a strict sense, the different WBEM resource identifiers used during paged retrieval of the resource each
 1405 represents a different resource, so that each subset returned by a paged retrieval operation would by a distinct
 1406 resource. However, that view makes it difficult to distinguish the entire resource from the subset resources retrieved.
 1407 Therefore, this document does not take that view and instead distinguishes between the subsets retrieved and the
 1408 (one) resource of which the subsets are retrieved.
 1409 NOTE: The use of HTTP range requests as defined in [RFC2616](#) has been considered and dismissed, because the
 1410 semantics of an ordered sequence of items that can be accessed by item number cannot be provided by
 1411 implementations that support the opaque server-defined enumeration context values mandated by [DSP0223](#).

1412 **8.4 Protocol payload elements**

1413 This clause defines the elements that are used in the payload of the messages of the operations defined
 1414 in clause 8. This clause defines these payload elements in a normative but abstract way. Documents
 1415 defining payload representation formats are expected to define how each of these payload elements is
 1416 represented.

1417 **8.4.1 NamespaceCollection payload element**

1418 A *NamespaceCollection* payload element represents a set of Namespace payload elements (see 8.4.2).
 1419 The members of the set depend on the context.

1420 A NamespaceCollection payload element includes the following properties:

1421 **Table 5 – Properties of NamespaceCollection payload element**

Property name	Generic type	Requirement	Description
namespaces	Namespace []	Mandatory	set of Namespace payload elements (see 8.4.2)

1422
 1423 A NamespaceCollection payload element includes the following links when used in response messages:

1424 **Table 6 – Links of NamespaceCollection payload element**

Link name	Absolute / Relative	Requirement	Targeted resource
self	Absolute	Mandatory	the represented namespace collection itself

1426 **8.4.2 Namespace payload element**

1427 A *Namespace* payload element represents a CIM namespace.

1428 There is no directly matching generic type representing a namespace payload element.

1429 A *Namespace* payload element includes the following properties:

1430 **Table 7 – Properties of Namespace payload element**

Property name	Generic type	Requirement	Description
name	string (see DSP0223)	Mandatory	namespace name (for example, "root/cimv2")

1431

1432 A *Namespace* payload element includes the following links when used in response messages:

1433 **Table 8 – Links of Namespace payload element**

Link name	Absolute / Relative	Requirement	Targeted resource
self	Absolute	Mandatory	the represented namespace itself
classes	Absolute or relative to self	Mandatory	class collection of all classes in the represented namespace
qualifiers	Absolute or relative to self	Mandatory	qualifier type collection of all qualifier types in the represented namespace
instancequery	Absolute or relative to self	Mandatory	instance query resource of the represented namespace

1434

1435 **8.4.3 ClassCollection payload element**

1436 A *ClassCollection* payload element represents a set of Class payload elements (see 8.4.4). The members
1437 of the set depend on the context.

1438 A *ClassCollection* payload element includes the following properties:

1439 **Table 9 – Properties of ClassCollection payload element**

Property name	Generic type	Requirement	Description
classes	Class []	Mandatory	set of Class payload elements (see 8.4.4)

1440

1441 A *ClassCollection* payload element includes the following links when used in response messages:

1442 **Table 10 – Links of ClassCollection payload element**

Link name	Absolute / Relative	Requirement	Targeted resource
self	Absolute	Mandatory	the represented class collection itself
namespace	Absolute or relative to self	Mandatory	namespace of the represented class collection

1443

1444 The "namespace" link of the ClassCollection payload element limits the class collection to classes from
 1445 the same namespace. In scenarios where association classes cross namespaces, this does not represent
 1446 a limitation; because all associated classes need to exist in both namespaces anyway.
 1447

1448 **8.4.4 Class payload element**

1449 A *Class* payload element represents a CIM class specification (that is, the definition of a class and its
 1450 elements).

1451 A Class payload element includes the following properties:

1452 **Table 11 – Properties of Class payload payload element**

Property name	Generic type	Requirement	Description
class	Class-Specification (see DSP0223)	Mandatory	class specification

1453

1454 A Class payload element includes the following links when used in response messages:

1455 **Table 12 – Links of Class payload element**

Link name	Absolute / Relative	Requirement	Targeted resource
self	Absolute	Mandatory	the represented class itself
namespace	Absolute or relative to self	Mandatory	namespace of the represented class
instances	Absolute or relative to self	Mandatory	instance collection of all instances of the represented class
method-invocation	Absolute or relative to self	Mandatory	class method invocation resource for all methods that can be invoked on the represented class
associators	Absolute or relative to self	Mandatory	class collection of all classes associated to the represented class
references	Absolute or relative to self	Mandatory	class collection of all classes referencing the represented class

1456

1457

1458 **8.4.5 InstanceCollection payload element**

1459 An *InstanceCollection* payload element represents a set of Instance payload elements (see 8.4.6). The
 1460 members of the set depend on the context.

1461 An InstanceCollection payload element includes the following properties:

1462 **Table 13 – Properties of InstanceCollection payload element**

Property name	Generic type	Requirement	Description
instances	Instance []	Mandatory	set of Instance payload elements (see 8.4.6)

1463

1464 An InstanceCollection payload element includes the following links when used in response messages:

1465 **Table 14 – Links of InstanceCollection payload element**

Link name	Absolute / Relative	Requirement	Targeted resource
self	Absolute	Mandatory	the represented instance collection itself
class	Absolute or relative to self	Conditional Exclusive	common superclass of the creation classes of the instances in the represented instance collection

1466

1467 Condition for the "class" link: The instance collection contains instances that have a common superclass.
 1468 If the condition is met, the "class" link shall be included; otherwise, it shall not be included.

1469 An example for an instance collection containing instances that may not have a common superclass is
 1470 the result set of traversing a ternary association to both far ends.

1471 **8.4.6 Instance payload element**

1472 An *Instance* payload element represents a CIM instance specification (that is, an instance including its
 1473 property values).

1474 An Instance payload element includes the following properties:

1475 **Table 15 – Properties of Instance payload element**

Property name	Generic type	Requirement	Description
class	creation class of Instance-Specification (see DSP0223)	Mandatory	name of creation class of the represented instance
properties	properties of Instance-Specification (see DSP0223)	Mandatory	properties of the represented instance, including their values

1476

1477 An Instance payload element includes the following links when used in response messages:

1478 **Table 16 – Links of Instance payload element**

Link name	Absolute / Relative	Requirement	Targeted resource
self	Absolute	Mandatory	the represented instance itself
class	Absolute or relative to self	Mandatory	creation class of the represented instance
method- invocation	Absolute or relative to self	Mandatory	instance method invocation resource for all methods that can be invoked on the represented instance
associators	Absolute or relative to self	Mandatory	instance collection of all instances associated to the represented instance
references	Absolute or relative to self	Mandatory	instance collection of all instances referencing the represented instance

1479
1480

1481 **8.4.7 QualifierTypeCollection payload element**

1482 A *QualifierTypeCollection* payload element represents a set of *QualifierType* payload elements (see
1483 8.4.8).

1484 A *QualifierTypeCollection* payload element includes the following properties:

1485 **Table 17 – Properties of QualifierTypeCollection payload element**

Property name	Generic type	Requirement	Description
qualifiertypes	QualifierType []	Mandatory	Set of <i>QualifierType</i> payload elements (see 8.4.8)

1486 A *QualifierTypeCollection* payload element includes the following links when used in response messages:

1487 **Table 18 – Links of QualifierTypeCollection payload element**

Link name	Absolute / Relative	Requirement	Targeted resource
self	Absolute	Mandatory	the represented qualifier type collection itself
namespace	Absolute or relative to self	Mandatory	namespace of the represented qualifier type collection

1488

1489 **8.4.8 QualifierType payload element**

1490 A *QualifierType* payload element represents a CIM qualifier type specification.

1491 A *QualifierType* payload element includes the following properties:

1492

Table 19 – Properties of QualifierType payload element

Property name	Generic type	Requirement	Description
type	string (see DSP0223)	Mandatory	Name of CIM datatype of the represented qualifier type specification
default	string (see DSP0223)	Mandatory	Default value of the represented qualifier type specification
scope	string [] (see DSP0223)	Mandatory	Set of scopes of the represented qualifier type specification
flavor	string [] (see DSP0223)	Mandatory	Set of flavors of the represented qualifier type specification

1493 A QualifierType payload element includes the following links when used in response messages:

1494

Table 20 – Links of QualifierType payload element

Link name	Absolute / Relative	Requirement	Targeted resource
self	Absolute	Mandatory	the represented qualifier type itself
namespace	Absolute or relative to self	Mandatory	namespace of the represented qualifier type

1495

1496 **8.4.9 MethodInvocationRequest payload element**

1497 A *MethodInvocationRequest* payload element represents the data used in a POST class method request
 1498 (see 8.12.1) or POST instance method request (see 8.17.1).

1499 A MethodInvocationRequest payload element includes the following properties:

1500

Table 21 – Properties of MethodInvocationRequest payload element

Property name	Generic type	Requirement	Description
name	MethodName (see DSP0223)	Mandatory	method name
parameters	Parameter-Value [] (see DSP0223)	Mandatory	set of input parameters

1501

1502 **8.4.10 MethodInvocationResponse payload element**

1503 A *MethodInvocationResponse* payload element represents the data used in a POST class method
 1504 response (see 8.12.1) or POST instance method response (see 8.17.1).

1505 A MethodInvocationResponse payload element includes the following properties:

1506

Table 22 – Properties of MethodInvocationResponse payload element

Property name	Generic type	Requirement	Description
name	MethodName (see DSP0223)	Mandatory	method name
returnvalue	ReturnValue (see DSP0223)	Mandatory	return value
parameters	Parameter-Value [] (see DSP0223)	Mandatory	set of output parameters

1507

8.4.11 InstanceModificationRequest payload element

1509 An *InstanceModificationRequest* payload element represents the data used in a PATCH instance request
 1510 (see 8.14.4). The InstanceModificationRequest payload element specifies modifications of property
 1511 values in a CIM instance.

1512 This version of this document defines property value replacement as the only type of property value
 1513 modification.

1514 Documents defining payload representations for the CIM-RS protocol need to ensure future extensibility
 1515 to other types of property value modifications, such as replacing, inserting or deleting elements in array
 1516 properties, or setting properties to their class-defined default value without specifying that value.

1517 An InstanceModificationRequest payload element includes the following properties:

1518

Table 23 – Properties of InstanceModificationRequest payload element

Property name	Generic type	Requirement	Description
replace	PropertyValue [] (see DSP0223)	Mandatory	set of properties whose values get replaced with new values

1519 The generic type PropertyReplacement is defined as follows:

1520

Table 24 – Properties of PropertyValue generic type

Property name	Generic type	Requirement	Description
name	PropertyName (see DSP0223)	Mandatory	name of the property to be replaced
value	string (see DSP0223)	Mandatory	new value for the property to be replaced, in its string representation as defined in DSP0004

1521

8.4.12 InstanceQueryRequest payload element

1523 An *InstanceQueryRequest* payload element represents the data used in a POST instance query request
 1524 (see 8.20.1).

1525 An InstanceQueryRequest payload element includes the following properties:

1526

Table 25 – Properties of InstanceQueryRequest payload element

Property name	Generic type	Requirement	Description
querystring	QueryString (see DSP0223)	Mandatory	query string in the query language identified by QueryLanguage
querylanguage	Query- Language (see DSP0223)	Mandatory	query language used in QueryString

1527 **8.4.13 ErrorResponse payload element**

1528 An *ErrorResponse* payload element represents an error response to any request.

1529 An *ErrorResponse* payload element includes the following properties:

1530

Table 26 – Properties of ErrorResponse payload element

Property name	Generic type	Requirement	Description
statuscode	integer (see DSP0223)	Optional	CIM status code
statusdescription	string (see DSP0223)	Optional	CIM status description
errors	Instance- Specification [] (see DSP0223)	Optional	set of embedded instances of class CIM_Error, each specifying an error message

1531

1532 **8.5 WBEM server and listener operations**

1533 This clause defines operations that target a WBEM server or WBEM listener as a whole (that is, without
1534 targeting resources within a WBEM server).

1535 **8.5.1 OPTIONS**

- 1536 **Purpose:** Retrieves WBEM server or WBEM listener options
- 1537 **HTTP method:** OPTIONS
- 1538 **Resource URI:** *
- 1539 **URI query parameters:** None
- 1540 **Request headers:** Host
- 1541 **Request payload:** None
- 1542 **Response headers:** Date, CIM-RS specific header fields as defined in Table 27 and Table 28
- 1543 **Response payload:** None
- 1544 **Generic operation:** N/A

1545 **Description:**

1546 An implementation of the HTTP OPTIONS method for the resource "*" shall return information about
 1547 the WBEM server that is targeted, using the response header fields described in Table 27:

1548 **Table 27 – Response header fields for OPTIONS to WBEM server**

Name of response header field	Requirement level	Defined in
CIMRS-Content-Types	Mandatory	9.4.5
CIMRS-Entity-Tagging-Feature (EXPERIMENTAL)	Mandatory	9.4.6
CIMRS-Paged-Retrieval-Feature	Mandatory	9.4.7
CIMRS-Filter-Query-Languages	Mandatory	9.4.8
CIMRS-Instance-Query-Languages	Mandatory	9.4.9

1549 An implementation of the HTTP OPTIONS method for the resource "*" shall return information about
 1550 the WBEM listener that is targeted, using the response header fields described in Table 28:

1551 **Table 28 – Response header fields for OPTIONS to WBEM listener**

Name of response header field	Requirement level	Defined in
CIMRS-Content-Types	Mandatory	9.4.5

1552 On success, a status code 200 "OK" shall be returned.

1553 **Example HTTP conversation with a WBEM server:**

```

1554 OPTIONS *
1555 Host: acme.com:5988
1556
1557 HTTP/1.1 200 OK
1558 Date: Wed, 29 Apr 2009 08:47:22 GMT
1559 CIMRS-Content-Types: application/vnd.dmtf.cimrs+json;version=1.0.0
1560 CIMRS-Etag-Feature: true
1561 CIMRS-Paged-Retrieval-Feature: true
    
```

1562 **8.6 Namespace collection operations**

1563 This clause defines operations that target a *namespace collection resource* (see 7.2.1).

1564 **8.6.1 GET namespace collection**

- 1565 **Purpose:** Retrieves the set of all namespaces in a WBEM server
- 1566 **HTTP method:** GET
- 1567 **Resource URI:** Namespace collection resource (defined in 7.2.1)
- 1568 **URI query parameters:** See Table 29
- 1569 **Request headers:** Host, Accept

- 1570 **Request payload:** None
- 1571 **Response headers:** Date, Content-Length, Content-Type
- 1572 **Response payload:** NamespaceCollection payload element (see 8.4.1) with links
- 1573 **Generic operation:** *GetClassInstancesWithPath* on class *CIM_Namespace* in the interop namespace of the WBEM server
- 1574

Table 29 – Query parameters for GET namespace collection

Name	Defined in
n (namespace)	7.3.13
ico (include class origin)	7.3.6
ip (included properties)	7.3.10
esbp (exclude subclass properties)	7.3.5

1576 **Description:**

1577 An implementation of the *GET namespace collection* operation shall conform to the behavior of the
 1578 generic operation *GetClassInstancesWithPath* defined in [DSP0223](#), when invoked with an
 1579 *EnumClassPath* parameter value that references the *CIM_Namespace* class in the interop
 1580 namespace of the WBEM server.

1581 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 1582 shall be returned, unless one of the following conditions applies:

- 1583 • For conditional requests where validators match, a status code 304 "Not Modified" and an
 1584 empty entity-body should be returned.
- 1585 • For range requests, a status code 206 "Partial Content" with the appropriate subset of
 1586 entities contained in the entity-body shall be returned.

1587 **Example HTTP conversation (using JSON):**

```

1588 GET /cimrs/namespaces
1589 Host: acme.com:5988
1590 Accept: application/vnd.dmtf.cimrs+json
1591
1592 HTTP/1.1 200 OK
1593 Date: Wed, 29 Apr 2009 08:47:22 GMT
1594 Content-Length: XXX
1595 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
1596 {
1597   "links": {
1598     "self": {
1599       "href": "http://acme.com:5988/cimrs/namespaces" }
1600   },
1601   "namespaces": {
1602     "root/cimv2": {
1603       "links": {
1604         "self": {
1605           "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2"},
    
```

```

1606     "classes": {
1607         "href": "classes"},
1608     "qualifiers": {
1609         "href": "qualifiers"},
1610     "instancequery": {
1611         "href": "instancequery"}
1612     }
1613 },
1614 "interop": {
1615     "links": {
1616         "self": {
1617             "href": "http://acme.com:5988/cimrs/namespaces/interop"},
1618         "classes": {
1619             "href": "classes"},
1620         "qualifiers": {
1621             "href": "qualifiers"},
1622         "instancequery": {
1623             "href": "instancequery"}
1624         }
1625     }
1626 }
1627 }

```

1628 NOTE: This example has an empty property list specified, so no property values are returned.

1629 8.7 Namespace operations

1630 This clause defines operations that target a *namespace resource* (see 7.2.2).

1631 8.7.1 GET namespace

1632	Purpose:	Retrieves a CIM namespace
1633	HTTP method:	GET
1634	Resource URI:	Namespace resource (defined in 7.2.2)
1635	URI query parameters:	See Table 30
1636	Request headers:	Host, Accept
1637	Request payload:	None
1638	Response headers:	Date, Content-Length, Content-Type, Etag
1639	Response payload:	Namespace payload element (see 8.4.2) with links
1640	Generic operation:	<i>GetInstance</i> on <i>CIM_Namespace</i> instance for that namespace, in the
1641		interop namespace of the WBEM server

1642

Table 30 – Query parameters for GET namespace

Name	Defined in
ico (include class origin)	7.3.6
ip (included properties)	7.3.10

1643

Description:

1644
1645
1646

An implementation of the *GET namespace* operation shall conform to the behavior of the generic operation *GetInstance* defined in [DSP0223](#), when invoked with an *InstancePath* parameter value that references the CIM_Namespace class in the interop namespace of the WBEM server.

1647
1648

On success, a status code 200 "OK" with the representation of the CIM_Namespace instance contained in the entity-body shall be returned, unless the following condition applies:

1649
1650

- For conditional requests where validators match, a status code 304 "Not Modified" and an empty entity-body should be returned.

1651

Example HTTP conversation (using JSON):

1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671

```
GET /cimrs/namespaces/root%2Fcimv2
Host: acme.com:5988
Accept: application/vnd.dmtf.cimrs+json

HTTP/1.1 200 OK
Date: Wed, 29 Apr 2009 08:47:22 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
"root/cimv2": {
  "links": {
    "self": {
      "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2"},
    "classes": {
      "href": "classes"},
    "qualifiers": {
      "href": "qualifiers"},
    "instancequery": {
      "href": "instancequery"}
  }
}
```

1672

8.8 Class collection operations

1673

This clause defines operations that target a *class collection resource* (see 7.2.3).

1674

8.8.1 GET class collection

1675
1676
1677

- Purpose:** Retrieves the set of classes in a class collection
- HTTP method:** GET
- Resource URI:** Class collection resource (defined in 7.2.3)

1678	URI query parameters:	See Table 31
1679	Request headers:	Host, Accept
1680	Request payload:	None
1681	Response headers:	Date, Content-Length, Content-Type
1682	Response payload:	ClassCollection payload element (see 8.4.3) with links
1683	Generic operation:	<i>GetTopClassesWithPath</i> , <i>GetSubClassesWithPath</i>

1684 **Table 31 – Query parameters for GET class collection**

Name	Defined in
c (class)	7.3.3
spc (superclass)	7.3.17
isbc (include subclasses)	7.3.12
iie (include inherited elements)	7.3.9
iq (include qualifiers)	7.3.11
ico (include class origin)	7.3.6

1685 **Description:**

1686 Depending on which query parameters are specified, the *GET class collection* operation shall
 1687 conform to generic operations defined in [DSP0223](#), as follows:

- 1688 • *GetTopClassesWithPath*, if the spc query parameter is not specified
- 1689 • *GetSuperClassesWithPath*, if the spc query parameter is specified

1690 If the spc query parameter is not specified, the *GET class collection* operation shall conform to the
 1691 generic operation *GetTopClassesWithPath* defined in [DSP0223](#), and the returned class collection
 1692 shall contain all classes in the namespace that have no superclasses.

1693 If the spc query parameter is specified, the *GET class collection* operation shall conform to the
 1694 generic operation *GetSubClassesWithPath* defined in [DSP0223](#), and the returned class collection
 1695 shall contain all classes in the namespace that have the specified class as a superclass.

1696 If the ip query parameter is specified with an empty property list, the returned class collection shall
 1697 consist of *self* URI values only.

1698 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 1699 shall be returned, unless one of the following conditions applies:

- 1700 • For conditional requests where validators match, a status code 304 "Not Modified" and an
 1701 empty entity-body should be returned.
- 1702 • For range requests, a status code 206 "Partial Content" with the appropriate subset of
 1703 entities contained in the entity-body shall be returned.

1704 **Example HTTP conversation (using JSON):**

1705 NOTE: This example specifies the spc query parameter.

```
1706 GET /cimrs/namespaces/root%2Fcimv2/classes?spc=CIM_Example
1707 Host: acme.com:5988
```

```

1708 Accept: application/vnd.dmtf.cimrs+json
1709
1710 HTTP/1.1 200 OK
1711 Date: Wed, 29 Apr 2009 08:47:22 GMT
1712 Content-Length: XXX
1713 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
1714 {
1715   "links": {
1716     "self": {
1717       "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes",
1718     "namespace": {
1719       "href": ".."}
1720   },
1721   "classes": {
1722     "CIM_SubClass1OfExample": {
1723       "links": {
1724         "self": {
1725           "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_S
1726 ubClass1OfExample"},
1727         "namespace": {
1728           "href": "../.."},
1729         "instances": {
1730           "href": "instances"},
1731         "methodinvocation": {
1732           "href": "methodinvocation"},
1733         "associators": {
1734           "href": "associators"},
1735         "references": {
1736           "href": "references"}
1737       },
1738       "superclass": "CIM_Example",
1739       "qualifiers": { . . . },
1740       "properties": { . . . }
1741     },
1742     "CIM_SubClass2OfExample": {
1743       "links": {
1744         "self": {
1745           "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_S
1746 ubClass2OfExample"},
1747         "namespace": {
1748           "href": "../.."},
1749         "instances": {
1750           "href": "instances"},
1751         "methodinvocation": {
1752           "href": "methodinvocation"},
1753         "associators": {
1754           "href": "associators"},
1755         "references": {
1756           "href": "references"}

```

```

1757     },
1758     "superclass": "CIM_Example",
1759     "qualifiers": { . . . },
1760     "properties": { . . . },
1761     "methods": { . . . ]
1762   }
1763 }
1764 }

```

1765 8.8.2 POST class collection

1766 **Purpose:** Creates a CIM class

1767 **HTTP method:** POST

1768 **Resource URI:** Class collection resource (defined in 7.2.3)

1769 **URI query parameters:** None

1770 **Request headers:** Host, Content-Length, Content-Type

1771 **Request payload:** Class payload element (see 8.4.4) without links

1772 **Response headers:** Date, Location

1773 **Response payload:** None

1774 **Generic operation:** *CreateClass*

1775 Description:

1776 The *POST class collection* operation shall conform to the *CreateClass* generic operation defined in
 1777 [DSP0223](#).

1778 On success, a status code 201 "Created" with the HTTP response header "Location" set to the
 1779 newly-created instance as indicated by its ClassPath shall be returned.

1780 Example HTTP conversation (using JSON):

```

1781 POST /cimrs/namespaces/root%2Fcimv2/classes
1782 Host: acme.com:5988
1783 Content-Length: XXX
1784 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
1785 "CIM_NewClass": {
1786   "superclass": "CIM_ManagedElement",
1787   "qualifiers": { . . . },
1788   "properties": { . . . },
1789   "methods": { . . . ]
1790 }
1791
1792 HTTP/1.1 201 Created
1793 Date: Wed, 29 Apr 2009 08:47:22 GMT
1794 Location: http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_NewClass

```

1795 **8.9 Class operations**

1796 This clause defines operations that target a *class resource* (see 7.2.4).

1797 The *ModifyClass* generic operation defined in [DSP0223](#) does not provide for modifying a class only
 1798 partially. For this reason, a *PATCH class* operation has not been defined in this document.

1799 **8.9.1 GET class**

- 1800 **Purpose:** Retrieves a CIM class
- 1801 **HTTP method:** GET
- 1802 **Resource URI:** Class resource (defined in 7.2.4)
- 1803 **URI query parameters:** See Table 32
- 1804 **Request headers:** Host, Accept
- 1805 **Request payload:** None
- 1806 **Response headers:** Date, Content-Length, Content-Type
- 1807 **Response payload:** Class payload element (see 8.4.4) with links
- 1808 **Generic operation:** *GetClass*

1809 **Table 32 – Query parameters for GET class**

Name	Defined in
iq (include qualifiers)	7.3.11
ico (include class origin)	7.3.6
ip (included properties)	7.3.10

1810 **Description:**

1811 The *GET class* operation shall conform to the *GetClass* generic operation defined in [DSP0223](#).

1812 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 1813 shall be returned, unless the following condition applies:

- 1814 • For conditional requests where validators match, a status code 304 "Not Modified" and an
 1815 empty entity-body should be returned.

1816 **Example HTTP conversation (using JSON):**

```

1817 GET /cimrs/namespaces/root%2Fcimv2/classes/CIM_RegisteredProfile
1818 Host: acme.com:5988
1819 Accept: application/vnd.dmtf.cimrs+json
1820
1821 HTTP/1.1 200 OK
1822 Date: Mon, 27 Apr 2009 17:02:09 GMT
1823 Content-Length: XXX
1824 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
1825 "CIM_RegisteredProfile": {
1826   "links": {
    
```

```

1827     "self": {
1828         "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_Regis
1829   teredProfile"},
1830     "namespace": {
1831         "href": "../.."},
1832     "instances": {
1833         "href": "instances"},
1834     "methodinvocation": {
1835         "href": "methodinvocation"},
1836     "associators": {
1837         "href": "associators"},
1838     "references": {
1839         "href": "references"}
1840   },
1841   "superclass": "CIM_ManagedElement",
1842   "qualifiers": { . . . },
1843   "properties": { . . . ]
1844 }

```

1845 **8.9.2 DELETE class**

- 1846 **Purpose:** Deletes a CIM class
- 1847 **HTTP method:** DELETE
- 1848 **Resource URI:** Class resource (defined in 7.2.4)
- 1849 **URI query parameters:** See Table 33
- 1850 **Request headers:** Host, Accept
- 1851 **Request payload:** None
- 1852 **Response headers:** Date
- 1853 **Response payload:** None
- 1854 **Generic operation:** *DeleteClass*

1855 **Table 33 – Query parameters for DELETE class**

Name	Defined in
dd (delete dependents)	7.3.3

1856 **Description:**

1857 The *DELETE* class operation shall conform to the *DeleteClass* generic operation defined in
 1858 [DSP0223](#).

1859 On success, a status code 204 "No Content" with an empty entity-body shall be returned.

1860 **Example HTTP conversation (using JSON):**

```

1861 DELETE /cimrs/namespaces/root%2Fcimv2/classes/CIM_RegisteredProfile
1862 Host: acme.com:5988

```

```

1863
1864 HTTP/1.1 204 No Content
1865 Date: Mon, 27 Apr 2009 17:02:09 GMT

```

1866 8.9.3 PUT class

1867 **Purpose:** Replaces a CIM class

1868 **HTTP method:** PUT

1869 **Resource URI:** Class resource (defined in 7.2.4)

1870 **URI query parameters:** None

1871 **Request headers:** Host, Content-Length, Content-Type

1872 **Request payload:** Class payload element (see 8.4.4) without links

1873 **Response headers:** Date

1874 **Response payload:** None

1875 **Generic operation:** *ModifyClass*

1876 Description:

1877 The *PUT class* operation shall conform to the *ModifyClass* generic operation defined in [DSP0223](#).

1878 On success, a status code 200 "OK" with the resource representation contained in the entity-body
1879 shall be returned.

1880 Example HTTP conversation (using JSON):

```

1881 PUT /cimrs/namespaces/root%2Fcimv2/classes/CIM_RegisteredProfile
1882 Host: acme.com:5988
1883 Content-Length: XXX
1884 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
1885 "CIM_RegisteredProfile": {
1886   "superclass": "CIM_ManagedElement",
1887   "qualifiers": { . . . },
1888   "properties": { . . . }
1889 }
1890
1891 HTTP/1.1 200 OK
1892 Date: Mon, 27 Apr 2009 17:02:09 GMT

```

1893 8.10 Class associator collection operations

1894 This clause defines operations that target a *class associator collection resource* (see 7.2.5).

1895 8.10.1 GET class associator collection

1896 **Purpose:** Retrieves the list of CIM classes that are associated with a given source
1897 class

1898 **HTTP method:** GET

- 1899 **Resource URI:** Class associator collection resource (defined in 7.2.5)
- 1900 **URI query parameters:** See Table 34
- 1901 **Request headers:** Host, Accept
- 1902 **Request payload:** None
- 1903 **Response headers:** Date, Content-Length, Content-Type
- 1904 **Response payload:** Class collection payload element (see 8.4.3) with links
- 1905 **Generic operation:** *GetAssociatedClassesWithPath*

1906 **Table 34 – Query parameters for GET class associator collection**

Name	Defined in
c (class)	7.3.3
rcn (referencing class name)	7.3.15
acn (associated class name)	7.3.1
srn (source role name)	7.3.18
arn (associated role name)	7.3.2
iq (include qualifiers)	7.3.11
ico (include class origin)	7.3.6
ip (included properties)	7.3.10

1907 **Description:**

1908 The *GET class associator collection* operation shall conform to the *GetAssociatedClassesWithPath*
 1909 generic operation defined in [DSP0223](#), with the following modified behavior:

- 1910 • The result of successive range requests may contain duplicate instances; duplicate
 1911 instances can be removed by the client by comparing the instances' *self* URI value as
 1912 defined by *Collection format*.
- 1913 • An empty *IncludedProperties* value corresponds to the *GetAssociatedClassPaths* generic
 1914 operation. The collection format returned in this case shall consist of *self* URI values only.

1915 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 1916 shall be returned, unless one of the following conditions applies:

- 1917 • For conditional requests where validators match, a status code 304 "Not Modified" and an
 1918 empty entity-body should be returned.
- 1919 • For range requests, a status code 206 "Partial Content" with the appropriate subset of
 1920 entities contained in the entity-body shall be returned.

1921 **Example HTTP conversation (using JSON):**

```

1922 GET /cimrs/namespaces/root%2Fcimv2/classes/CIM_SourceExample/associators
1923 Host: acme.com:5988
1924 Accept: application/vnd.dmtf.cimrs+json
1925
1926 HTTP/1.1 200 OK
    
```

```

1927 Date: Wed, 29 Apr 2009 08:47:22 GMT
1928 Content-Length: XXX
1929 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
1930 {
1931   "links": {
1932     "self": {
1933       "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_Sourc
1934 eExample/associators"},
1935     "namespace": {
1936       "href": "../.."}
1937   },
1938   "classes": {
1939     "CIM_AssociatedExample": {
1940       "links": {
1941         "self": {
1942           "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_A
1943 ssociatedExample"},
1944         "namespace": {
1945           "href": "../.."},
1946         "instances": {
1947           "href": "instances"},
1948         "methodinvocation": {
1949           "href": "methodinvocation"},
1950         "associators": {
1951           "href": "associators"},
1952         "references": {
1953           "href": "references"}
1954       },
1955       "qualifiers": { . . . },
1956       "properties": { . . . }
1957     },
1958     . . . # more classes
1959   }
1960 }

```

1961 8.11 Class reference collection operations

1962 This clause defines operations that target a *class reference collection resource* (see 7.2.6).

1963 8.11.1 GET class reference collection

1964 **Purpose:** Retrieves the list of CIM association classes that reference a given
 1965 source class

1966 **HTTP method:** GET

1967 **Resource URI:** Class reference collection resource (defined in 7.2.6)

1968 **URI query parameters:** See Table 35

1969 **Request headers:** Host, Accept

- 1970 **Request payload:** None
- 1971 **Response headers:** Date, Content-Length, Content-Type
- 1972 **Response payload:** Class collection payload element (see 8.4.3) with links
- 1973 **Generic operation:** *GetReferencingClassesWithPath*

1974 **Table 35 – Query parameters for GET class reference collection**

Name	Defined in
c (class)	7.3.3
rcn (referencing class name)	7.3.15
acn (associated class name)	7.3.1
srn (source role name)	7.3.18
arn (associated role name)	7.3.2
iq (include qualifiers)	7.3.11
ico (include class origin)	7.3.6
ip (included properties)	7.3.10

1975 **Description:**

1976 The *GET class reference collection* operation shall conform to the *GetReferencingClassesWithPath*
 1977 generic operation defined in [DSP0223](#), with the following modified behavior:

- 1978 • The result of successive range requests may contain duplicate classes; duplicate classes
 1979 can be removed by the client by comparing the class *self* URI value as defined by
 1980 *Collection format*.
- 1981 • An empty *IncludedProperties* value corresponds to the *GetReferencingClassPaths* generic
 1982 operation. The *Collection* format returned in this case shall consist of *self* URI values only.

1983 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 1984 shall be returned, unless one of the following conditions applies:

- 1985 • For conditional requests where validators match, a status code 304 "Not Modified" and an
 1986 empty entity-body should be returned.
- 1987 • For range requests, a status code 206 "Partial Content" with the appropriate subset of
 1988 entities contained in the entity-body shall be returned.

1989 **Example HTTP conversation (using JSON):**

```

1990 GET /cimrs/namespaces/root%2Fcimv2/classes/CIM_SourceExample/references
1991 Host: acme.com:5988
1992 Accept: application/vnd.dmtf.cimrs+json
1993
1994 HTTP/1.1 200 OK
1995 Date: Wed, 29 Apr 2009 08:47:22 GMT
1996 Content-Length: XXX
1997 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
1998 {
    
```

```

1999     "links": {
2000         "self": {
2001             "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_Sourc
2002 eExample/references"},
2003         "namespace": {
2004             "href": "../.."}
2005     },
2006     "classes": {
2007         "CIM_AssociationExample": {
2008             "links": {
2009                 "self": {
2010                     "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_A
2011 ssociationExample"},
2012                 "namespace": {
2013                     "href": "../.."},
2014                 "instances": {
2015                     "href": "instances"},
2016                 "methodinvocation": {
2017                     "href": "methodinvocation"},
2018                 "associators": {
2019                     "href": "associators"},
2020                 "references": {
2021                     "href": "references"}
2022             },
2023             "qualifiers": { . . . },
2024             "properties": { . . . }
2025         },
2026         . . . # more classes
2027     }
2028 }

```

2029 8.12 Class method operations

2030 This clause defines operations that target a *class method resource* (see 7.2.7).

2031 8.12.1 POST class method

2032	Purpose:	Invokes a (static) CIM method on a class
2033	HTTP method:	POST
2034	Resource URI:	Class method invocation resource (defined in 7.2.7)
2035	URI query parameters:	None
2036	Request headers:	Host, Accept, Content-Length, Content-Type
2037	Request payload:	MethodInvocationRequest payload element (see 8.4.9)
2038	Response headers:	Date, Content-Length, Content-Type
2039	Response payload:	MethodInvocationResponse payload element (see 8.4.10)

2040 **Generic operation:** *InvokeStaticMethod*

2041 **Description:**

2042 The *POST* class method operation shall conform to the *InvokeStaticMethod* generic operation defined
2043 in [DSP0223](#).

2044 On success, a status code of either 200 "OK" or 202 "Accepted" shall be returned.

2045 **Example HTTP conversation (using JSON):**

```

2046 POST /cimrs/namespaces/root%2Fcimv2/classes/CIM_RegisteredProfile/methodinvocation
2047 Host: acme.com:5988
2048 Accept: application/vnd.dmtf.cimrs+json
2049 Content-Length: XXX
2050 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2051 "StaticMethod": {
2052   "parameters": {
2053     "Parm1": "Some input value"
2054   }
2055 }
2056
2057 HTTP/1.1 200 OK
2058 Date: Wed, 29 Apr 2009 08:47:22 GMT
2059 Content-Length: XXX
2060 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2061 "StaticMethod": {
2062   "returnvalue": 0,
2063   "parameters": {
2064     "Parm2": "Some output value"
2065   }
2066 }

```

2067 8.13 Instance collection operations

2068 This clause defines operations that target an *instance collection resource* (see 7.2.8).

2069 8.13.1 GET instance collection

2070 **Purpose:** Retrieves the list of CIM instances of a given CIM class

2071 **HTTP method:** GET

2072 **Resource URI:** Instance collection resource (defined in 7.2.8)

2073 **URI query parameters:** See Table 36

2074 **Request headers:** Host, Accept

2075 **Request payload:** None

2076 **Response headers:** Date, Content-Length, Content-Type

2077 **Response payload:** InstanceCollection payload element (see 8.4.5) with links

2078 **Generic operation:** *GetClassInstancesWithPath*

2079 **Table 36 – Query parameters for GET instance collection**

Name	Defined in
fql (filter query language)	7.3.6
fqs (filter query string)	7.3.7
ico (include class origin)	7.3.6
ip (included properties)	7.3.10
esbp (exclude subclass properties)	7.3.5

2080 **Description:**

2081 The *GET instance collection* operation shall conform to the *GetClassInstancesWithPath* generic
 2082 operation defined in [DSP0223](#), with the following modified behavior:

- 2083 • The result of successive range requests may contain duplicate instances; duplicate
 2084 instances can be removed by the client by comparing the instances' *self* URI value as
 2085 defined by *Collection format*.
- 2086 • An empty *IncludedProperties* value corresponds to the *GetClassInstancePaths* generic
 2087 operation. The Collection format returned in this case shall consist of *self* URI values only.

2088 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 2089 shall be returned, unless one of the following conditions applies:

- 2090 • For conditional requests where validators match, a status code 304 "Not Modified" and an
 2091 empty entity-body should be returned.
- 2092 • For range requests, a status code 206 "Partial Content" with the appropriate subset of
 2093 entities contained in the entity-body shall be returned.

2094 **Example HTTP conversation (using JSON):**

```

2095 GET /cimrs/namespaces/root%2Fcimv2/classes/CIM_RegisteredProfile/instances
2096 Host: acme.com:5988
2097 Accept: application/vnd.dmtf.cimrs+json
2098
2099 HTTP/1.1 200 OK
2100 Date: Wed, 29 Apr 2009 08:47:22 GMT
2101 Content-Length: XXX
2102 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2103 {
2104   "links": {
2105     "self": {
2106       "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_Sourc
2107 eExample/instances"},
2108     "class": {
2109       "href": ".."}
2110   },
2111   "instances": [
2112     {
2113       "links": {
```

```

2114     "self": {
2115         "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_R
2116 registeredProfile/instances/DMTF%3AFan%3A1.1.0"},
2117     "class": {
2118         "href": "../.."},
2119     "methodinvocation": {
2120         "href": "methodinvocation"},
2121     "associators": {
2122         "href": "associators"},
2123     "references": {
2124         "href": "references"}
2125     },
2126     "class": "CIM_RegisteredProfile",
2127     "properties": {
2128         "InstanceID": "DMTF:Fan:1.1.0",
2129         "RegisteredName": "Fan",
2130         "RegisteredOrganization": 2,
2131         "RegisteredVersion": "1.1.0",
2132         . . . # more properties
2133     }
2134 },
2135 . . . # more instances
2136 ]
2137 }

```

2138 8.13.2 POST instance collection

2139	Purpose:	Creates a CIM instance
2140	HTTP method:	POST
2141	Resource URI:	Instance collection resource (defined in 7.2.8)
2142	URI query parameters:	None
2143	Request headers:	Host, Content-Length, Content-Type
2144	Request payload:	Instance payload element (see 8.4.6) without links
2145	Response headers:	Date, Location
2146	Response payload:	None
2147	Generic operation:	<i>CreateInstance</i>

2148 Description:

2149 The *POST instance collection* operation shall conform to the *CreateInstance* generic operation
 2150 defined in [DSP0223](#).

2151 On success, a status code 201 "Created" with the HTTP response header "Location" set to the newly
 2152 created instance as indicated by its InstancePath shall be returned.

2153 **Example HTTP conversation (using JSON):**

```

2154 POST /cimrs/namespaces/root%2Fcimv2/classes/CIM_RegisteredProfile/instances
2155 Host: acme.com:5988
2156 Content-Length: XXX
2157 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2158 {
2159   "class": "CIM_RegisteredProfile",
2160   "properties": {
2161     "RegisteredName": "Fan",
2162     "RegisteredOrganization": 2,
2163     "RegisteredVersion": "1.1.0",
2164     . . . # more properties
2165   }
2166 }
2167
2168 HTTP/1.1 201 Created
2169 Date: Wed, 29 Apr 2009 08:47:22 GMT
2170 Location: http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_Registered
2171 Profile/instances/DMTF%3AFan%3A1.1.0
    
```

2172 NOTE: The key property InstanceID is not provided in the request, since key property values are determined
 2173 by the server. In this example, the InstanceID value has been determined by the server from the provided values
 2174 of the RegisteredOrganization, RegisteredName, and RegisteredVersion properties.

2175 **8.14 Instance operations**

2176 This clause defines operations that target an *instance resource* (see 7.2.9).

2177 **8.14.1 GET instance**

- 2178 **Purpose:** Retrieves a CIM instance
- 2179 **HTTP method:** GET
- 2180 **Resource URI:** Instance resource (defined in 7.2.9)
- 2181 **URI query parameters:** See Table 37
- 2182 **Request headers:** Host, Accept
- 2183 **Request payload:** None
- 2184 **Response headers:** Date, Content-Length, Content-Type, Etag
- 2185 **Response payload:** Instance payload element (see 8.4.6)
- 2186 **Generic operation:** *GetInstance*

2187 **Table 37 – Query parameters for GET instance**

Name	Defined in
ico (include class origin)	7.3.6
ip (included properties)	7.3.10

2188 Description:

2189 The *GET instance* operation shall conform to the *GetInstance* generic operation defined in [DSP0223](#).

2190 On success, a status code 200 "OK" with the resource representation contained in the entity-body
2191 shall be returned, unless the following condition applies:

- 2192 • For conditional requests where validators match, a status code 304 "Not Modified" and an
2193 empty entity-body should be returned.

2194 Example HTTP conversation (using JSON):

```

2195 GET /cimrs/namespaces/root%2Fcimv2/classes/CIM_RegisteredProfile/instances/DMTF%3AF
2196 an%3A1.1.0
2197 Host: acme.com:5988
2198 Accept: application/vnd.dmtf.cimrs+json
2199
2200 HTTP/1.1 200 OK
2201 Date: Wed, 29 Apr 2009 08:47:22 GMT
2202 Content-Length: XXX
2203 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2204 {
2205   "links": {
2206     "self": {
2207       "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_Regis
2208   teredProfile/instances/DMTF%3AFan%3A1.1.0"},
2209     "class": {
2210       "href": "../.."},
2211     "methodinvocation": {
2212       "href": "methodinvocation"},
2213     "associators": {
2214       "href": "associators"},
2215     "references": {
2216       "href": "references"}
2217   },
2218   "class": "CIM_RegisteredProfile",
2219   "properties": {
2220     "InstanceID": "DMTF:Fan:1.1.0",
2221     "RegisteredName": "Fan",
2222     "RegisteredOrganization": 2,
2223     "RegisteredVersion": "1.1.0",
2224     . . . # more properties
2225   }
2226 }

```

2227 8.14.2 DELETE instance

2228 **Purpose:** Deletes a CIM instance

2229 **HTTP method:** DELETE

2230 **Resource URI:** Instance resource (defined in 7.2.9)

2231	URI query parameters:	None
2232	Request headers:	Host
2233	Request payload:	None
2234	Response headers:	Date
2235	Response payload:	None
2236	Generic operation:	<i>DeleteInstance</i>

2237 **Description:**

2238 The *DELETE instance* operation shall conform to the *DeleteInstance* generic operation defined in
 2239 [DSP0223](#).

2240 On success, a status code 204 "No Content" with an empty entity-body shall be returned.

2241 **Example HTTP conversation (using JSON):**

```

2242 DELETE /cimrs/namespaces/root%2Fcimv2/classes/CIM_RegisteredProfile/instances/DMTF%
2243 3AFan%3A1.1.0
2244 Host: acme.com:5988
2245
2246 HTTP/1.1 204 No Content
2247 Date: Wed, 29 Apr 2009 08:47:22 GMT
  
```

2248 **8.14.3 PUT instance**

2249	Purpose:	Replaces a CIM instance
2250	HTTP method:	PUT
2251	Resource URI:	InstancePath (defined in 7.2.9)
2252	URI query parameters:	None
2253	Request headers:	Host, Content-Length, Content-Type, If-Match (EXPERIMENTAL)
2254	Request payload:	Instance payload element (see 8.4.6) without links
2255	Response headers:	Date
2256	Response payload:	None
2257	Generic operation:	<i>ModifyInstance</i> with <i>IncludedProperties</i> being Null

2258 **Description:**

2259 The *PUT instance* operation shall conform to the *ModifyInstance* generic operation defined in
 2260 [DSP0223](#), when its *IncludedProperties* parameter is Null. The *ModifyInstance* generic operation will
 2261 modify all properties of the CIM instance that are modifiable, in this case.

2262 The instance payload element does not need to specify all properties exposed by the creation class
 2263 of the instance. The values of modifiable properties that are not specified in the instance payload
 2264 element shall be set to the class-defined default value of the property, or to Null if no such default
 2265 value is defined.

2266 NOTE: This behavior for properties that are requested to be changed but not specified in the instance payload
 2267 element is consistent with [DSP0200](#). In contrast, [DSP0223](#) requires that the property is set to Null in this case,
 2268 even when a non-Null default value for the property is defined in the class.
 2269

2270 The values of non-modifiable properties shall not be changed by the WBEM server. *PUT instance*
 2271 operations that specify non-modifiable properties in the instance payload element with a value that is
 2272 different from their current value shall be rejected.

2273 EXPERIMENTAL

2274 In addition, the *PUT instance* operation shall reject the modification if an If-Match header field is
 2275 provided, and the entity tag provided as its value does not match the current entity tag (that is, the
 2276 value of the Generation property) of the resource.

2277 EXPERIMENTAL

2278 On success, either of the following status codes shall be returned:

- 2279 • 200 "OK". with an empty entity-body
- 2280 • 202 "Accepted" with an empty entity-body

2281 Example HTTP conversation (using JSON):

```

2282 PUT /cimrs/namespaces/root%2Fcimv2/classes/CIM_RegisteredProfile/instances/DMTF%3AF
2283 an%3A1.1.0
2284 Host: acme.com:5988
2285 Content-Length: XXX
2286 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2287 {
2288   "class": " CIM_RegisteredProfile",
2289   "properties": {
2290     "RegisteredName": "Fan",
2291     "RegisteredOrganization": 2,
2292     "RegisteredVersion": "1.1.1",
2293     "Caption": "A changed caption"
2294   }
2295 }
2296
2297 HTTP/1.1 200 OK
2298 Date: Wed, 29 Apr 2009 08:47:22 GMT
  
```

2299 NOTE: In this example, it is assumed that all provided properties are modifiable. The modifiable properties not
 2300 provided are set to their class-defined default values or to Null. The value of the InstanceID key property remains
 2301 unchanged, since key properties are never modifiable.

2302 8.14.4 PATCH instance

2303 **Purpose:** Modifies properties of a CIM instance

2304 **HTTP method:** PATCH

2305 **Resource URI:** Instance resource (defined in 7.2.9)

2306 **URI query parameters:** None

2307	Request headers:	Host, Content-Length, Content-Type, If-Match (EXPERIMENTAL)
2308	Request payload:	InstanceModificationRequest payload element (see 8.4.11)
2309	Response headers:	Date
2310	Response payload:	None
2311	Generic operation:	<i>ModifyInstance</i> with <i>IncludedProperties</i> being non-Null
2312	Description:	
2313		The <i>PATCH instance</i> operation shall conform to the <i>ModifyInstance</i> generic operation defined in
2314		DSP0223 , when its <i>IncludedProperties</i> parameter is non-Null. In this case, the <i>ModifyInstance</i>
2315		generic operation will modify only the properties that are specified in its <i>IncludedProperties</i>
2316		parameter and that are modifiable.
2317		The InstanceModificationRequest payload element (see 8.4.11) specifies the requested changes for
2318		the instance.
2319		The InstanceModificationRequest payload element does not need to specify requested changes for
2320		all properties exposed by the creation class of the instance. Properties for which no changes are
2321		specified in the InstanceModificationRequest payload element shall be left unchanged.

2322 EXPERIMENTAL

2323 In addition, the *PATCH instance* operation shall reject the modification if an If-Match header field is
 2324 provided, and the entity tag provided as its value does not match the current entity tag (that is, the
 2325 value of the Generation property) of the resource.

2326 EXPERIMENTAL

2327 On success, either of the following status codes shall be returned:

- 2328 • 200 "OK". with an empty entity-body
- 2329 • 202 "Accepted" with an empty entity-body

2330 Example HTTP conversation (using JSON):

```

2331 PATCH /cimrs/namespaces/root%2Fcimv2/classes/CIM_RegisteredProfile/instances/DMTF%3
2332 AFan%3A1.1.0
2333 Host: acme.com:5988
2334 Content-Length: XXX
2335 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2336 {
2337   "replace": {
2338     "Caption": "A changed caption"
2339   }
2340 }
2341
2342 HTTP/1.1 200 OK
2343 Date: Wed, 29 Apr 2009 08:47:22 GMT

```

2344 NOTE: In this example, only the value of the Caption property is requested to be changed. It is assumed to be
 2345 modifiable. All other property values remain unchanged, since they are not requested to be changed.

2346 **8.15 Instance associator collection operations**

2347 This clause defines operations that target an *instance associator collection resource* (see 7.2.10).

2348 **8.15.1 GET instance associator collection**

- 2349 **Purpose:** Retrieves the list of CIM instances that are associated with a given
- 2350 source instance
- 2351 **HTTP method:** GET
- 2352 **Resource URI:** Instance associator collection resource (defined in 7.2.10)
- 2353 **URI query parameters:** See Table 38
- 2354 **Request headers:** Host, Accept
- 2355 **Request payload:** None
- 2356 **Response headers:** Date, Content-Length, Content-Type
- 2357 **Response payload:** InstanceCollection payload element (see 8.4.5)
- 2358 **Generic operation:** *GetAssociatedInstancesWithPath*

2359 **Table 38 – Query parameters for GET instance associator collection**

Name	Defined in
fql (filter query language)	7.3.6
fqs (filter query string)	7.3.7
rcn (referencing class name)	7.3.15
acn (associated class name)	7.3.1
srn (source role name)	7.3.18
arn (associated role name)	7.3.2
ico (include class origin)	7.3.6
ip (included properties)	7.3.10
esbp (exclude subclass properties)	7.3.5

2360 **Description:**

2361 The *GET instance associator collection* operation shall conform to the
 2362 *GetAssociatedInstancesWithPath* generic operation defined in [DSP0223](#), with the following modified
 2363 behavior:

- 2364 • The result of successive range requests may contain duplicate instances; duplicate
 2365 instances can be removed by the client by comparing the instances' *self* URI value as
 2366 defined by *Collection format*.
- 2367 • An empty *IncludedProperties* value corresponds to the *GetAssociatedInstancePaths*
 2368 generic operation. The Collection format returned in this case shall consist of *self* URI
 2369 values only.

2370 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 2371 shall be returned, unless one of the following conditions applies:

- 2372 – For conditional requests where validators match, a status code 304 "Not Modified" and an empty
 2373 entity-body should be returned.
- 2374 – For range requests, a status code 206 "Partial Content" with the appropriate subset of entities
 2375 contained in the entity-body shall be returned.

2376 **Example HTTP conversation (using JSON):**

```

2377 GET /cimrs/namespaces/root%2Fcimv2/classes/CIM_SourceExample/instances/acme1/associ
2378 ators
2379 Host: acme.com:5988
2380 Accept: application/vnd.dmtf.cimrs+json
2381
2382 HTTP/1.1 200 OK
2383 Date: Wed, 29 Apr 2009 08:47:22 GMT
2384 Content-Length: XXX
2385 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2386 {
2387   "links": {
2388     "self": {
2389       "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_Sourc
2390 eExample/instances/acme1/associators"},
2391     "class": {
2392       "href": "../.."}
2393   },
2394   "instances": [
2395     {
2396       "links": {
2397         "self": {
2398           "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_A
2399 ssociatedExample/instances/acme2"},
2400         "class": {
2401           "href": "../.."},
2402         "methodinvocation": {
2403           "href": "methodinvocation"},
2404         "associators": {
2405           "href": "associators"},
2406         "references": {
2407           "href": "references"}
2408       },
2409       "class": "CIM_AssociatedExample",
2410       "properties": {
2411         "InstanceID": "acme2",
2412         "Caption": "An example associated instance",
2413         . . . # more properties
2414       }
2415     },
2416     . . . # more instances
2417   ]

```

2418 }

2419 8.16 Instance reference collection operations

2420 This clause defines operations that target an *instance reference collection resource* (see 7.2.11).

2421 8.16.1 GET instance reference collection

2422 **Purpose:** Retrieves the list of CIM association instances that reference a given
2423 source instance

2424 **HTTP method:** GET

2425 **Resource URI:** Instance reference collection resource (defined in 7.2.11)

2426 **URI query parameters:** See Table 39

2427 **Request headers:** Host, Accept

2428 **Request payload:** None

2429 **Response headers:** Date, Content-Length, Content-Type

2430 **Response payload:** InstanceCollection payload element (see 8.4.5)

2431 **Generic operation:** *GetReferencingInstancesWithPath*

2432 **Table 39 – Query parameters for GET instance reference collection**

Name	Defined in
fql (filter query language)	7.3.6
fqs (filter query string)	7.3.7
rcn (referencing class name)	7.3.15
can (associated class name)	7.3.1
srn (source role name)	7.3.18
arn (associated role name)	7.3.2
ico (include class origin)	7.3.6
ip (included properties)	7.3.10
esbp (exclude subclass properties)	7.3.5

2433 Description:

2434 The *GET instance reference collection* operation shall conform to the
2435 *GetReferencingInstancesWithPath* generic operation defined in [DSP0223](#), with the following modified
2436 behavior:

- 2437 • The result of successive range requests may contain duplicate instances; duplicate
2438 instances can be removed by the client by comparing the instances' *self* URI value as
2439 defined by *Collection format*.
- 2440 • An empty *IncludedProperties* value corresponds to the *GetClassInstancePaths* generic
2441 operation. The *Collection format* returned in this case shall consist of *self* URI values only.

2442 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 2443 shall be returned, unless one of the following conditions applies:

- 2444 • For conditional requests where validators match, a status code 304 "Not Modified" and an
 2445 empty entity-body should be returned.
- 2446 • For range requests, a status code 206 "Partial Content" with the appropriate subset of
 2447 entities contained in the entity-body shall be returned.

2448 Example HTTP conversation (using JSON):

```

2449 GET /cimrs/namespaces/root%2Fcimv2/classes/CIM_SourceExample/instances/acme1/refere
2450 nces
2451 Host: acme.com:5988
2452 Accept: application/vnd.dmtf.cimrs+json
2453
2454 HTTP/1.1 200 OK
2455 Date: Wed, 29 Apr 2009 08:47:22 GMT
2456 Content-Length: XXX
2457 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2458 {
2459   "links": {
2460     "self": {
2461       "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_Sourc
2462 eExample/instances/acme1/references"},
2463     "class": {
2464       "href": "../.."}
2465   },
2466   "instances": [
2467     {
2468       "links": {
2469         "self": {
2470           "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes/CIM_A
2471 ssociationExample/instances/http%3A%2F%2Facme.com%3A5988%2Fcimrs%2Fnamespaces%2Froo
2472 t%252Fcimv2%2Fclasses%2FCIM_SourceExample%2Finstances%2Facme1,http%3A%2F%2Facme.com
2473 %3A5988%2Fcimrs%2Fnamespaces%2Froot%252Fcimv2%2Fclasses%2FCIM_AssociatedExample%2Fi
2474 nstances%2Facme2"},
2475         "class": {
2476           "href": "../.."},
2477         "methodinvocation": {
2478           "href": "methodinvocation"},
2479         "associators": {
2480           "href": "associators"},
2481         "references": {
2482           "href": "references"}
2483       },
2484       "class": "CIM_AssociationExample",
2485       "properties": {
2486         "InstanceID": "acme1,acme2",
2487         "Caption": "An example association instance referencing the given
2488 instance",
2489         . . . # more properties
2490       }
    }
  ]
}

```

```

2491     },
2492     . . . # more instances
2493   ]
2494 }

```

2495 NOTE: In this example, the key properties of the CIM_AssociationExample association class are its two
 2496 reference properties, as usually in the CIM Schema. The key property values are again WBEM resource
 2497 identifiers, which need to be percent-escaped when used as instance key values in the WBEM resource identifier
 2498 of the association instance.

2499 8.17 Instance method operations

2500 This clause defines operations that target an *instance method resource* (see 7.2.12).

2501 8.17.1 POST instance method

2502 **Purpose:** Invokes a (static or non-static) CIM method on a CIM instance

2503 **HTTP method:** POST

2504 **Resource URI:** Instance method invocation resource (defined in 7.2.12)

2505 **URI query parameters:** None

2506 **Request headers:** Host, Accept, Content-Length, Content-Type

2507 **Request payload:** MethodInvocationRequest payload element (see 8.4.9)

2508 **Response headers:** Date, Content-Length, Content-Type

2509 **Response payload:** MethodInvocationResponse payload element (see 8.4.10)

2510 **Generic operation:** *InvokeMethod*

2511 Description:

2512 The *POST instance method* operation shall conform to the *InvokeMethod* generic operation defined
 2513 in [DSP0223](#).

2514 On success, a status code of either 200 "OK" or 202 "Accepted" shall be returned.

2515 Example HTTP conversation (using JSON):

```

2516 POST /cimrs/namespaces/root%2Fcimv2/classes/CIM_Example/instances/acme1/methodinvoc
2517 ation
2518 Host: acme.com:5988
2519 Accept: application/vnd.dmtf.cimrs+json
2520 Content-Length: XXX
2521 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2522 "SomeMethod": {
2523   "parameters": {
2524     "Parm1": "Some input data"
2525   }
2526 }
2527
2528 HTTP/1.1 200 OK

```

```

2529 Date: Wed, 29 Apr 2009 08:47:22 GMT
2530 Content-Length: XXX
2531 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2532 "SomeMethod": {
2533     "returnValue": 42,
2534     "parameters": {
2535         "Parm2": "Some output data"
2536     }
2537 }
    
```

2538 **8.18 Qualifier type collection operations**

2539 This clause defines operations that target a *qualifier type collection resource* (see 7.2.13).

2540 **8.18.1 GET qualifier type collection**

- 2541 **Purpose:** Retrieves the list of CIM qualifier types in a CIM namespace
- 2542 **HTTP method:** GET
- 2543 **Resource URI:** Qualifier type collection resource (defined in 7.2.13)
- 2544 **URI query parameters:** See Table 40
- 2545 **Request headers:** Host, Accept
- 2546 **Request payload:** None
- 2547 **Response headers:** Date, Content-Length, Content-Type
- 2548 **Response payload:** QualifierTypeCollection payload element (see 8.4.7) with links
- 2549 **Generic operation:** *EnumerateQualifierTypesWithPath*

2550 **Table 40 – Query parameters for GET qualifier type collection**

Name	Defined in
q (qualifier)	7.3.15

2551 **Description:**

2552 The *GET qualifier type collection* operation shall conform to the *EnumerateQualifierTypesWithPath*
 2553 generic operation defined in [DSP0223](#), with the following modified behavior:

- 2554 • The result of successive range requests may contain duplicate qualifier types; duplicate
 2555 qualifier types can be removed by the client by comparing the qualifier types' *self* URI
 2556 value as defined by *Collection format*.

2557 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 2558 shall be returned, unless one of the following conditions applies:

- 2559 • For conditional requests where validators match, a status code 304 "Not Modified" and an
 2560 empty entity-body should be returned.
- 2561 • For range requests, a status code 206 "Partial Content" with the appropriate subset of
 2562 entities contained in the entity-body shall be returned.

2563 **Example HTTP conversation (using JSON):**

```

2564 GET /cimrs/namespaces/root%2Fcimv2/qualifiers
2565 Host: acme.com:5988
2566 Accept: application/vnd.dmtf.cimrs+json
2567
2568 HTTP/1.1 200 OK
2569 Date: Mon, 27 Apr 2009 17:02:09 GMT
2570 Content-Length: XXX
2571 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2572 {
2573   "links": {
2574     "self": {
2575       "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/qualifiers"},
2576     "namespace": {
2577       "href": ".."}
2578   },
2579   "qualifiertypes": {
2580     "Counter": {
2581       "links": {
2582         "self": {
2583           "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/qualifiers/Co
2584 unter"},
2585         "namespace": {
2586           "href": "../.."}
2587       },
2588       "type": "boolean",
2589       "default": false,
2590       "scope": [ "property", "method", "parameter" ],
2591       "flavor": [ "ToSubclass" ]
2592     },
2593     . . . # more qualifier types
2594   }
2595 }

```

2596 **8.18.2 POST qualifier type collection**

2597	Purpose:	Creates a CIM qualifier type
2598	HTTP method:	POST
2599	Resource URI:	Qualifier type collection resource (defined in 7.2.13)
2600	URI query parameters:	None
2601	Request headers:	Host, Content-Length, Content-Type
2602	Request payload:	QualifierType payload element (see 8.4.8) without links
2603	Response headers:	Date, Location
2604	Response payload:	None

2605 **Generic operation:** *CreateQualifierType*

2606 **Description:**

2607 The *POST qualifier type collection* operation shall conform to the *CreateQualifierType* generic
2608 operation defined in [DSP0223](#).

2609 On success, a status code 201 "Created" with the HTTP response header "Location" set to the
2610 newly-created instance as indicated by its *QualifierTypePath* shall be returned.

2611 **Example HTTP conversation (using JSON):**

```

2612 POST /cimrs/namespaces/root%2Fcimv2/qualifiers
2613 Host: acme.com:5988
2614 Content-Length: XXX
2615 Content-Type: application/vnd.dmtf.cimrs+json
2616 "NewQualifier": {
2617   "type": "string",
2618   "default": null,
2619   "scope": [ "property" ],
2620   "flavor": [ "Restricted" ]
2621 }
2622
2623 HTTP/1.1 201 Created
2624 Date: Mon, 27 Apr 2009 17:02:09 GMT
2625 Location: http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/qualifiers/NewQualifie
2626 r

```

2627 8.19 Qualifier type operations

2628 This clause defines operations that target a *qualifier type resource* (see 7.2.14).

2629 The *ModifyQualifierType* generic operation defined in [DSP0223](#) does not provide for modifying a qualifier
2630 type only partially. For this reason, a *PATCH qualifier type* operation has not been defined in this
2631 document.

2632 8.19.1 GET qualifier type

2633 **Purpose:** Retrieves a CIM qualifier type

2634 **HTTP method:** GET

2635 **Resource URI:** Qualifier type resource (defined in 7.2.14)

2636 **URI query parameters:** None

2637 **Request headers:** Host, Accept

2638 **Request payload:** None

2639 **Response headers:** Date, Content-Length, Content-Type

2640 **Response payload:** QualifierType payload element (see 8.4.8) with links

2641 **Generic operation:** *GetQualifierType*

2642 **Description:**

2643 The *GET qualifier type* operation shall conform to the *GetQualifierType* generic operation defined in
 2644 [DSP0223](#).

2645 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 2646 shall be returned, unless the following condition applies:

- 2647 • For conditional requests where validators match, a status code 304 "Not Modified" and an
 2648 empty entity-body should be returned.

2649 **Example HTTP conversation (using JSON):**

```

2650 GET /cimrs/namespaces/root%2Fcimv2/qualifiers/Alias
2651 Host: acme.com:5988
2652 Accept: application/vnd.dmtf.cimrs+json
2653
2654 HTTP/1.1 200 OK
2655 Date: Mon, 27 Apr 2009 17:02:09 GMT
2656 Content-Length: XXX
2657 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2658 "Alias": {
2659   "links": {
2660     "self": {
2661       "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/qualifiers/Alias"
2662     },
2663     "namespace": {
2664       "href": "../.."
2665     },
2666     "type": "string",
2667     "default": null,
2668     "scope": [ "property" ],
2669     "flavor": [ "Restricted" ]
2670   }

```

2671 **8.19.2 DELETE qualifier type**

2672	Purpose:	Deletes a CIM qualifier type
2673	HTTP method:	DELETE
2674	Resource URI:	Qualifier type resource (defined in 7.2.14)
2675	URI query parameters:	None
2676	Request headers:	Host
2677	Request payload:	None
2678	Response headers:	Date
2679	Response payload:	None
2680	Generic operation:	<i>DeleteQualifierType</i>

2681 **Description:**

2682 The *DELETE qualifier type* operation shall conform to the *DeleteQualifierType* generic operation
 2683 defined in [DSP0223](#).

2684 On success, a status code 204 "No Content" with an empty entity-body shall be returned.

2685 **Example HTTP conversation (using JSON):**

```
2686 DELETE /cimrs/namespaces/root%2Fcimv2/qualifiers/Alias
2687 Host: acme.com:5988
2688
2689 HTTP/1.1 204 No Content
2690 Date: Mon, 27 Apr 2009 17:02:09 GMT
```

2691 **8.19.3 PUT qualifier type**

2692 **Purpose:** Replaces a CIM qualifier type

2693 **HTTP method:** PUT

2694 **Resource URI:** Qualifier type resource (defined in 7.2.14)

2695 **URI query parameters:** None

2696 **Request headers:** Host, Content-Length, Content-Type

2697 **Request payload:** QualifierType payload element (see 8.4.8) without links

2698 **Response headers:** Date

2699 **Response payload:** None

2700 **Generic operation:** *ModifyQualifierType*

2701 **Description:**

2702 The *PUT qualifier type* operation shall conform to the *ModifyQualifierType* generic operation defined
 2703 in [DSP0223](#).

2704 On success, a status code 200 "OK" with the resource representation contained in the entity-body
 2705 shall be returned.

2706 **Example HTTP conversation (using JSON):**

```
2707 PUT /cimrs/namespaces/root%2Fcimv2/qualifiers/Alias
2708 Host: acme.com:5988
2709 Content-Length: XXX
2710 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2711 "Alias": {
2712   "type": "uint8",
2713   "default": null,
2714   "scope": [ "property" ],
2715   "flavor": [ "Restricted" ]
2716 }
2717
2718 HTTP/1.1 200 OK
```

2719 Date: Mon, 27 Apr 2009 17:02:09 GMT

2720 8.20 Instance query operations

2721 This clause defines operations that target an *instance query resource* (see 7.2.15).

2722 EXPERIMENTAL

2723 8.20.1 POST instance query

2724 **Purpose:** Execute a query for instances on a CIM namespace and return the query
2725 result

2726 **HTTP method:** POST

2727 **Resource URI:** Instance query resource (defined in 7.2.15)

2728 **URI query parameters:** None

2729 **Request headers:** Host, Content-Length, Content-Type, Accept

2730 **Request payload:** InstanceQueryRequest payload element (see 8.4.12)

2731 **Response headers:** Date, Content-Length, Content-Type

2732 **Response payload:** InstanceCollection payload element (see 8.4.5) for the instances
2733 representing the query result, without links

2734 **Generic operation:** An assumed future *ExecuteQuery* operation with the abilities of the
2735 ExecQuery operation defined in [DSP0200](#)

2736 Description:

2737 The *POST instance query* operation shall conform to the (future) *ExecuteQuery* generic operation
2738 defined in [DSP0223](#), with the following added behavior:

- 2739 • The query shall return an instance collection
- 2740 • The query shall not modify any CIM entities

2741 On success, a status code 200 "OK" with the resource representation contained in the entity-body
2742 shall be returned, unless the following condition applies:

- 2743 • For conditional requests where validators match, a status code 304 "Not Modified" and an
2744 empty entity-body should be returned.

2745 Example HTTP conversation (using JSON):

```
2746 POST /cimrs/namespaces/root%2Fcimv2/instancequery
2747 Host: acme.com:5988
2748 Accept: application/vnd.dmtf.cimrs+json
2749 Content-Length: XXX
2750 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2751 {
2752   "querystring": "SELECT OBJECTPATH(CIM_StorageExtent) AS Path, ElementName
2753                 FROM CIM_StorageExtent",
2754   "querylanguage": "DMTF:CQL"
```

```

2755 }
2756
2757 HTTP/1.1 200 OK
2758 Content-Length: XXX
2759 Content-Type: application/vnd.dmtf.cimrs+json;version=1.0.0
2760 {
2761   "instances": [
2762     {
2763       "class": "TBD",
2764       "properties": {
2765         "Path": "/namespaces/root%2Fcimv2/classes/CIM_StorageExtent/instances/acme5
2766 0",
2767         "ElementName": "Example instance"
2768       }
2769     },
2770     . . . # more instances
2771   ]
2772 }

```

2773 **EXPERIMENTAL**

2774 8.21 Functionality without specific operations

2775 This clause describes functionality that does not have specific operations defined.

2776 8.21.1 Subscription and other indication related functions

2777 Subscription for indications and other indication-related functions (such as retrieving and managing
 2778 indication filters, filter collections, listener destinations or indication services) can be performed using
 2779 other operations defined in this document, if the *Indications Profile* ([DSP1054](#)) is implemented.

2780 8.22 Potential future operations

2781 This clause describes potential future operations, in order to get feedback on them.

2782 8.22.1 Indication delivery

2783 **Purpose:** Delivers an indication to a WBEM listener. Details of this operation are to
 2784 be defined.

2785 9 Usage of HTTP and HTTPS

2786 9.1 HTTP and HTTPS version requirements

2787 WBEM clients, WBEM servers, and WBEM listeners shall support the use of HTTP for the CIM-RS
2788 protocol. The following applies for this case:

- 2789 • Version 1.1 of HTTP shall be supported as defined in [RFC2616](#).
- 2790 • Version 1.0 of HTTP shall not be supported.

2791 WBEM clients, WBEM servers, and WBEM listeners should support the use of HTTPS for the CIM-RS
2792 protocol. If they do, the following applies:

- 2793 • HTTPS shall be supported as defined in [RFC2818](#).
- 2794 • The secure sockets layer used with HTTPS shall be TLS 1.1 as defined in [RFC4346](#); in addition,
2795 the secure sockets layer should be TLS 1.2 as defined in [RFC5246](#).
- 2796 • SSL 2.0, SSL 3.0, or TLS 1.0 (also known as SSL 3.1) shall not be supported as the secure
2797 sockets layer used with HTTPS because of security issues in these versions.
- 2798 • Within HTTPS, version 1.1 of HTTP shall be supported as defined in [RFC2616](#).

2799 NOTE 1 HTTPS should not be confused with *Secure HTTP* defined in RFC2660.

2800 NOTE 2 [RFC2818](#) describes the use of TLS 1.0 and higher but not the use of SSL 2.0 or 3.0.

2801 NOTE 3 [RFC2818](#) describes the use of HTTP/1.0 and HTTP/1.1 within HTTPS.

2802 9.2 Authentication requirements

2803 This clause describes requirements and considerations for authentication between WBEM clients, WBEM
2804 servers, and WBEM listeners. Specifically, authentication happens from WBEM clients to WBEM servers
2805 for operation messages, and from WBEM servers to WBEM listeners for indication delivery messages.

2806 9.2.1 Operating without authentication

2807 WBEM clients, WBEM servers, and WBEM listeners may support operating without the use of
2808 authentication.

2809 This may be acceptable in environments such as physically secured networks or between components on
2810 the same operating system.

2811 9.2.2 HTTP basic authentication

2812 HTTP basic authentication provides a rudimentary level of authentication, with the major weakness that
2813 the client password is part of the HTTP headers in unencrypted form.

2814 WBEM clients, WBEM servers, and WBEM listeners may support HTTP basic authentication as defined in
2815 [RFC2617](#).

2816 HTTP basic authentication may be acceptable in environments such as physically secured networks,
 2817 between components on the same operating system, or when the messages are encrypted by using
 2818 HTTPS.

2819 **9.2.3 HTTP digest authentication**

2820 HTTP digest authentication verifies that both parties share a common secret without having to send that
 2821 secret in the clear. Thus, it is more secure than HTTP basic authentication.

2822 WBEM clients, WBEM servers, and WBEM listeners should support HTTP digest authentication as
 2823 defined in [RFC2617](#).

2824 **9.2.4 Other authentication mechanisms**

2825 WBEM clients, WBEM servers, and WBEM listeners may support authentication mechanisms not covered
 2826 by [RFC2617](#). One example of such a mechanism is public key certificates as defined in [X.509](#).

2827 **9.3 Message encryption**

2828 Encryption of HTTP messages can be supported by the use of HTTPS and its secure sockets layer.

2829 Requirements for the use of HTTPS and its underlying secure sockets are defined in 9.1.

2830 It is important to understand that authentication and encryption of messages are separate issues:
 2831 Encryption of messages requires the use of HTTPS, while the authentication mechanisms defined in 9.2
 2832 can be used with both HTTP and HTTPS.

2833 **9.4 HTTP header fields**

2834 This clause describes the use of HTTP header fields within the CIM-RS protocol, and it defines additional
 2835 CIM-RS specific header fields.

2836 **9.4.1 Accept**

2837 The rules for the Accept request header field defined in [RFC2616](#) apply. This clause defines additional
 2838 constraints on its use.

2839 The Accept request header field may be provided on the request message of any operation that may
 2840 return a response payload.

2841 If provided, it shall specify all MIME media types identifying valid CIM-RS payload representations that
 2842 are supported by the WBEM client or WBEM server, using the following format (defined in ABNF):

2843 `Accept = "Accept" WS ":" 1*(WS media-type [WS accept-params])`

2844 where `media-type` is a MIME media type identifying a valid CIM-RS payload representation, and
 2845 `accept-params` is defined in [RFC2616](#).

2846 If an Accept request header field is provided, WBEM servers shall use one of the valid CIM-RS payload
 2847 representations identified in the Accept request header field for the response payload.

2848 If none of the CIM-RS payload representations identified in the Accept request header field is supported
 2849 by the WBEM server or WBEM listener, it shall return 406 "not acceptable".

2850 NOTE: [RFC2616](#) recommends returning 406 "not acceptable" in this case, but it does not require it.

2851 WBEM servers and WBEM listeners shall ignore any invalid `media-type` values, as well as the use of
2852 media ranges such as `"/*`.

2853 Support for `accept-params` is optional.

2854 If no Accept request header field is provided, WBEM servers and WBEM listeners may use any valid CIM-
2855 RS payload representation for the response payload.

2856 WBEM clients and WBEM servers are not required to identify the same set of CIM-RS payload
2857 representations in an Accept request header field as in previous requests.

2858 When multiple CIM-RS payload representations are indicated in an Accept request header field, WBEM
2859 servers and WBEM listeners are not required to use the same CIM-RS payload representation as in
2860 previous responses.

2861 **9.4.2 Content-Type**

2862 The rules for the Content-Type entity header field defined in [RFC2616](#) apply. This clause defines
2863 additional constraints on its use.

2864 As defined in [RFC2616](#), the Content-Type entity header field shall be provided on the request message of
2865 any operation that passes a request payload and on the response message of any operation that returns
2866 a response payload.

2867 The Content-Type entity header field shall specify the MIME media type identifying the CIM-RS payload
2868 representation that is used for the payload.

2869 **EXPERIMENTAL**

2870 **9.4.3 Etag**

2871 The rules for the Etag response header field defined in [RFC2616](#) apply. This clause defines additional
2872 constraints on its use.

2873 The Etag response header field shall be provided in any response by a WBEM server for which all of the
2874 following criteria are satisfied:

- 2875 • The entity tagging feature (see 8.3.1) is implemented by the WBEM server.
- 2876 • The response is for GET instance (see 8.14.1) or GET namespace (see 8.7.1).
- 2877 • The CIM instance has a non-Null value of its (string typed) Generation property.

2878 In this case, the Etag response header field shall be specified using the following format (defined in
2879 ABNF):

```
2880 Etag = "Etag" WS ":" generation
```

2881 where `generation` is the value of the Generation property of the CIM instance represented in the
2882 response.

2883 Otherwise, the Etag response header field shall not be provided by a WBEM server.

2884 The Etag response header field shall not be provided in any responses from a WBEM listener.

2885 **EXPERIMENTAL**

2886 EXPERIMENTAL**2887 9.4.4 If-Match**

2888 The rules for the If-Match request header field defined in [RFC2616](#) apply. This clause defines additional
2889 constraints on its use.

2890 The If-Match request header field shall be provided in any request by a WBEM client for which all of the
2891 following criteria are satisfied:

- 2892 • The entity tagging feature (see 8.3.1) is implemented by the WBEM client.
- 2893 • The request is for a PUT instance (see 8.14.3).
- 2894 • The new instance supplied in the PUT request is based on an instance that was retrieved earlier
2895 and that had the Etag header field (see 9.4.3) set.

2896 In this case, the If-Match request header field shall be specified using the following format (defined in
2897 ABNF):

2898 `If-Match = "If-Match" WS ":" generation`

2899 where `generation` is the value of the Etag header field of the CIM instance that is the basis for the
2900 modification.

2901 Otherwise, the If-Match request header field shall not be provided in any request by a WBEM client.

2902 The If-Match request header field shall not be provided in any request by WBEM server to a WBEM
2903 listener.

2904 EXPERIMENTAL

2905 9.4.5 CIMRS-Content-Types

2906 The CIMRS-Content-Types response header field identifies the CIM-RS payload representations
2907 supported by the WBEM server and WBEM listener by specifying their MIME media types, using the
2908 following format (defined in ABNF):

2909 `CIMRS-Content-Types = "CIMRS-Content-Types" WS ":" 1*(WS media-type)`

2910 where `media-type` is a MIME media type identifying a valid CIM-RS payload representation.

2911 See 8.5.1 for its use in the HTTP OPTIONS method.

2912 EXPERIMENTAL**2913 9.4.6 CIMRS-Entity-Tagging-Feature**

2914 The CIMRS-Entity-Tagging-Feature response header field indicates with a boolean value whether the
2915 entity tagging feature (see 8.3.1) is implemented by the WBEM server, using the following format (defined
2916 in ABNF):

2917 `CIMRS-Etag-Feature = "CIMRS-Entity-Tagging-Feature" WS ":" ("true" / "false")`

2918 See 8.5.1 for its use in the HTTP OPTIONS method targeting a WBEM server.

2919 EXPERIMENTAL

2920 **9.4.7 CIMRS-Paged-Retrieval-Feature**

2921 The CIMRS-Paged-Retrieval-Feature response header field indicates with a boolean value whether
2922 paged retrieval (see 8.3.2) is implemented by the WBEM server, using the following format (defined in
2923 ABNF):

```
2924 CIMRS-Paged-Retrieval-Feature = "CIMRS-Paged-Retrieval-Feature" WS ":" ( "true" /  
2925 "false" )
```

2926 See 8.5.1 for its use in the HTTP OPTIONS method targeting a WBEM server.

2927 **9.4.8 CIMRS-Filter-Query-Languages**

2928 The CIMRS-Filter-Query-Languages response header field indicates which query languages are
2929 supported for the fql (filter query language, see 7.3.6) query parameter, using the following format
2930 (defined in ABNF):

```
2931 CIMRS-Filter-Query-Languages = "CIMRS-Filter-Query-Languages" WS ":" WS [ query-  
2932 language *( WS "," WS query-language ) ]
```

2933 Where query-language is a string that uniquely identifies a query language.

2934 See 8.5.1 for its use in the HTTP OPTIONS method targeting a WBEM server.

2935 **9.4.9 CIMRS-Instance-Query-Languages**

2936 The CIMRS-Instance-Query-Languages response header field indicates which query languages are
2937 supported for the query provided with a POST instance query operation (see 8.20.1), using the following
2938 format (defined in ABNF):

```
2939 CIMRS-Instance-Query-Languages = "CIMRS-Instance-Query-Languages" WS ":" WS [  
2940 query-language *( WS "," WS query-language ) ]
```

2941 Where query-language is a string that uniquely identifies a query language.

2942 See 8.5.1 for its use in the HTTP OPTIONS method targeting a WBEM server.

2943 10 Payload representation

2944 Payload representations for the content of HTTP messages used by the CIM-RS protocol are expected to
2945 be described in separate documents.

2946 This clause defines requirements for such CIM-RS payload representation descriptions.

2947 10.1 Media type

2948 A CIM-RS payload representation description shall define a MIME media type by which that payload
2949 representation can be uniquely identified within the set of all payload representations defined for the CIM-
2950 RS protocol.

2951 This media type shall not use media ranges (that is, the asterisk character "*") for its type or subtype
2952 fields.

2953 The combination of type and subtype fields of this media type shall change when the payload
2954 representation changes incompatibly.

2955 It is recommended to encode the full version of the payload representation in this media type, preferably
2956 as a media type parameter named "version".

2957 A CIM-RS payload representation description shall define the rules used to determine whether a payload
2958 representation identified by a particular media type (including a version) is compatible to a consumer that
2959 understands a particular media type (including a version).
2960

2961 10.2 Payload element representations

2962 A CIM-RS payload representation description shall define a representation for each payload element
2963 defined in this document (see 8.4).

2964 Generic properties of payload elements may be represented in any way in the payload representation.
2965 The generic property name stated in the subclauses of in 8.4 does not need to be retained in the payload
2966 representation.

2967 For example, in a JSON representation of a Namespace payload element (see 8.4.2), all of the following
2968 options would be valid for representing the "name" generic property for a namespace named "root/cimv2":

- 2969 • as the JSON object name:

```
2970 "root/cimv2": {
2971     # remaining generic properties and links
2972 }
```

- 2973 • as a JSON attribute with the same name as the generic property:

```
2974 {
2975     "name": "root/cimv2",
2976     # remaining generic properties and links
2977 }
```

- 2978 • as a JSON attribute with a different name as the generic property:

```
2979 {
```

```
2980     "namespace-name": "root/cimv2",
2981     # remaining generic properties and links
2982 }
```

2983 Links of payload elements may be represented in any way in the payload representation that retains the
2984 link name and target.

2985 For example, in a JSON representation of a Namespace payload element (see 8.4.2), all of the following
2986 options would be valid for representing the links:

- 2987 • as a JSON attribute named "links" which is a JSON array of JSON objects named with the link
2988 name that has the link target as an attribute:

```
2989 {
2990     "links": {
2991         "self": {
2992             "href": "http://. . ." # URI of the "self" link
2993         },
2994         # remaining links
2995     },
2996     # generic properties
2997 }
```

- 2998 • as a JSON attribute named "links" which is a JSON array of unnamed JSON objects that have
2999 both link name and link target as attributes:

```
3000 {
3001     "links": [
3002         {
3003             "rel": "self",
3004             "href": "http://. . ." # URI of the "self" link
3005         },
3006         # remaining links
3007     ],
3008     # generic properties
3009 }
```

3010

3011

3012 **ANNEX A**
3013 (informative)

3014
3015 **Known payload representations**
3016

3017 This annex lists the CIM-RS payload representations known at the time of release of this document.

3018 **Table 41 – Known CIM-RS payload representations**

Name	Underlying format	Defined in
CIM-RS Binding to JSON	Javascript Object Notation (JSON)	DSP-IS0202

3019 It is expected that an XML-based payload representation for CIM-RS will be defined in the future.

ANNEX B (informative)

3020
3021
3022
3023
3024

Examples for structure of resource URIs

3025 This annex describes examples on how the resource URIs for a CIM-RS implementation could be
3026 structured.

3027 **B.1 Example using CIM object types as URI segments**

3028 In this example, the guiding principle is that the terms for CIM object types (namespace, class, instance,
3029 qualifier type) and the terms for certain related resources (associator, reference, etc.) are used as the
3030 segments of the resource URIs. The term identifying the type of object is used in its plural form as one
3031 URI segment, followed by a segment that identifies the instance of that type of object within the context of
3032 the higher segments. Such tuples of segments are repeated along certain hierarchies of these CIM object
3033 types. In the last of such tuples in a URI, the second segment may be omitted to address the collection of
3034 the respective resources, instead of a single resource.

3035 **B.1.1 Namespace collection resource**

3036 A WBEM resource identifier addressing a namespace collection resource needs to conform to the
3037 structure defined in 7.2.1.

3038 **B.1.2 Namespace resource**

3039 A WBEM resource identifier addressing a namespace resource could conform to the WBEM-resource-
3040 identifier ABNF rule (see 7.1.1) with the following additional rules:

```
3041 path-absolute = NamespacePath
3042
3043 NamespacePath = NamespaceCollectionPath "/" NamespaceName
```

3044 Where:

- 3045 • NamespaceName is the percent-encoded name of the CIM namespace (see 7.1.2 for percent-
3046 encoding rules).
- 3047 NOTE Besides other possible transformations, applying these percent-encoding rules causes any slash
3048 ("/") characters in the namespace name to be represented as the string "%2F".
- 3049 • NamespaceCollectionPath is defined in 7.2.1.

3050 Examples:

- 3051 • `http://acme.com:5988/cimrs/namespaces/interop`

3052 This resource identifier addresses the "interop" namespace of the WBEM server at port 5988 of
3053 host "acme.com", using HTTP.

- 3054 • `/cimrs/namespaces/myns`

3055 This resource identifier addresses the "myns" namespace of a WBEM server whose host, port,
3056 and scheme are known by other means.

- 3057 • <https://acme.com/cimrs/namespaces/root%2Fcimv2>

3058 This resource identifier addresses the "root/cimv2" namespace of the WBEM server at the
3059 default port 443 of host "acme.com", using HTTPS.

3060 **B.1.3 Class collection resource**

3061 A WBEM resource identifier addressing a class collection resource could conform to the WBEM-resource-
3062 identifier ABNF rule (see 7.1.1) with the following additional rules:

```
3063 path-absolute = ClassCollectionPath
3064
3065 ClassCollectionPath = NamespacePath "/"classes"
```

3066 Example:

- 3067 • /cimrs/namespaces/myns/classes

3068 This resource identifier addresses the collection of all classes in the "myns" namespace of a
3069 WBEM server whose host, port, and scheme are known by other means.

3070 **B.1.4 Class resource**

3071 A WBEM resource identifier addressing a class resource could conform to the WBEM-resource-identifier
3072 ABNF rule (see 7.1.1) with the following additional rules:

```
3073 path-absolute = ClassPath
3074
3075 ClassPath = ClassCollectionPath "/" ClassName
```

3076 Where:

- 3077 • ClassName is the percent-encoded name of the CIM class (including its schema prefix). (See
3078 7.1.2 for percent-encoding rules.)

3079 Example:

- 3080 • /cimrs/namespaces/myns/classes/CIM_ManagedElement

3081 This resource identifier addresses the "CIM_ManagedElement" class in the "myns" namespace
3082 of a WBEM server whose host, port, and scheme are known by other means.

3083 **B.1.5 Class associator collection resource**

3084 A WBEM resource identifier addressing a class associator collection resource could conform to the
3085 WBEM-resource-identifier ABNF rule (see 7.1.1) with the following additional rules:

```
3086 path-absolute = ClassAssociatorCollectionPath
3087
3088 ClassAssociatorCollectionPath = ClassPath "/associators"
```

3089 Example:

- 3090 • /cimrs/namespaces/myns/classes/CIM_ManagedElement/associators

3091 This resource identifier addresses the collection of all classes associated with the
3092 "CIM_ManagedElement" class in the "myns" namespace of a WBEM server whose host, port,
3093 and scheme are known by other means.

3094 **B.1.6 Class reference collection resource**

3095 A WBEM resource identifier addressing a class reference collection resource could conform to the
3096 WBEM-resource-identifier ABNF rule (see 7.1.1) with the following additional rules:

```
3097 path-absolute = ClassReferenceCollectionPath  
3098  
3099 ClassReferenceCollectionPath = ClassPath "/"references"
```

3100 Example:

- 3101 • /cimrs/namespaces/root%2Fcimv2/classes/CIM_ManagedElement/references

3102 This resource identifier addresses the collection of all classes referencing the
3103 "CIM_ManagedElement" class in the "root/cimv2" namespace of a WBEM server whose host,
3104 port, and scheme are known by other means.

3105 **B.1.7 Class method invocation resource**

3106 A WBEM resource identifier addressing a class method invocation resource could conform to the WBEM-
3107 resource-identifier ABNF rule (see 7.1.1) with the following additional rules:

```
3108 path-absolute = ClassMethodInvocationPath  
3109  
3110 ClassMethodInvocationPath = ClassPath "/"methodinvocation"
```

3111 Example:

- 3112 • /cimrs/namespaces/root%2Fcimv2/classes/CIM_Example/methodinvocation

3113 This resource identifier addresses the method invocation point exposed by the "CIM_Example"
3114 class in the "root/cimv2" namespace of a WBEM server whose host, port, and scheme are
3115 known by other means.

3116 **B.1.8 Instance collection resource**

3117 A WBEM resource identifier addressing an instance collection resource could conform to the WBEM-
3118 resource-identifier ABNF rule (see 7.1.1) with the following additional rules:

```
3119 path-absolute = InstanceCollectionPath  
3120  
3121 InstanceCollectionPath = ClassPath "/"instances"
```

3122 Example:

- 3123 • /cimrs/namespaces/myns/classes/CIM_ManagedElement/instances

3124 This resource identifier addresses the collection of all instances in the "myns" namespace that
3125 have a creation class of "CIM_ManagedElement".

3126 **B.1.9 Instance resource**

3127 A WBEM resource identifier addressing an instance resource could conform to the WBEM-resource-
3128 identifier ABNF rule (see 7.1.1) with the following additional rules:

```
3129 path-absolute = InstancePath  
3130  
3131 InstancePath = InstanceCollectionPath "/" KeyList
```

```

3132
3133 KeyList =
3134     OrdinaryKeyList / ; for instances of ordinary classes
3135     AssociationKeyList ; for instances of association classes
3136
3137 OrdinaryKeyList =
3138     OrdinaryKeyPropertyValue *( "," OrdinaryKeyPropertyValue )
3139     ; list is ordered by property names
3140
3141 AssociationKeyList =
3142     AssociationKeyValue *( "," AssociationKeyValue )
3143     ; list is ordered by property names
3144
3145 AssociationKeyValue =
3146     OrdinaryKeyPropertyValue / ; for ordinary key properties
3147     "(" ReferenceValue ")" ; for key references
3148
3149 ReferenceValue = [ InstanceCollectionPath "/" ] KeyList
3150     ; of the referenced instance

```

3151 Where:

- 3152 • KeyList is OrdinaryKeyList for instances of ordinary classes and AssociationKeyList
3153 for instances of association classes.
- 3154 • The key value lists in OrdinaryKeyList and AssociationKeyList are ordered by their
3155 property names, using binary ordering based on the data octets of the UTF-8 representation of
3156 the normalized Unicode characters of the property name.
- 3157 • OrdinaryKeyPropertyValue is the percent-encoded string representation of the value of an
3158 ordinary key property. (See 7.1.2 for percent-encoding rules, and see [DSP0004](#) for the string
3159 representations of CIM data types.)
- 3160 • ReferenceValue uses the KeyList value of the referenced instance.

3161 NOTE: This is the same KeyList value that would be used in a resource identifier addressing that
3162 instance directly.

3163 NOTE: The use of parentheses around ReferenceValue in the AssociationKeyValue rule enables
3164 references to reference other association instances.

3165 NOTE: This version of the document does not yet define rules for the use of InstanceCollectionPath
3166 in ReferenceValue.

3167
3168

3169 Examples:

- 3170 • /cimrs/namespaces/myns/classes/ACME_System/42,Dent%2CArthur

3171 Assuming ACME_System exposes key properties Name (string) and Answer (uint32), this
3172 resource identifier addresses the ACME_System instance in the "myns" namespace that has a
3173 Name value of "Dent,Arthur" and an Answer value of 42.

3174 This example shows that the order of key values in the resource identifier is in the order of
3175 property names.

- 3176 The comma in the Name value has been percent-encoded, as defined in 7.1.2.
- 3177 • /cimrs/namespaces/myns/classes/ACME_LogicalDevice/(babel%2Dfish),(42,D
3178 ent%2CArthur)
- 3179 Assuming the ACME_LogicalDevice association class exposes key references System (REF
3180 ACME_System) and Device (REF ACME_Device), where ACME_System is defined as in the
3181 previous example and ACME_Device exposes a key property Name (string), this resource
3182 identifier addresses the ACME_LogicalDevice instance in the "myns" namespace that
3183 associates the ACME_System instance with key values 42 and "Dent,Arthur" (see previous
3184 example) and the ACME_Device instance with a Name value of "babel-fish".
- 3185 This example shows that the order of key values in the resource identifier is in the order of
3186 property names, at each level of the referencing hierarchy.
- 3187 The hash character in the Name value the ACME_Device instance has been percent-encoded,
3188 as defined in 7.1.2.

3189 **B.1.10 Instance associator collection resource**

3190 A WBEM resource identifier addressing an instance associator collection resource could conform to the
3191 WBEM-resource-identifier ABNF rule (see 7.1.1) with the following additional rules:

```
3192 path-absolute = InstanceAssociatorCollectionPath
3193
3194 InstanceAssociatorCollectionPath = InstancePath "/"associators"
```

3195 Example:

- 3196 • /cimrs/namespaces/myns/classes/CIM_ManagedElement/instances/acme,1/ass
3197 ociators
- 3198 This resource identifier addresses the collection of all instances associated with the
3199 "CIM_ManagedElement" instance in the "myns" namespace that is identified by the key list
3200 "acme,1"

3201 **B.1.11 Instance reference collection resource**

3202 A WBEM resource identifier addressing an instance reference collection resource could conform to the
3203 WBEM-resource-identifier ABNF rule (see 7.1.1) with the following additional rules:

```
3204 path-absolute = InstanceReferenceCollectionPath
3205
3206 InstanceReferenceCollectionPath = InstancePath "/"references"
```

3207 Example:

- 3208 • /cimrs/namespaces/myns/classes/CIM_ManagedElement/instances/acme,1/ref
3209 erences
- 3210 This resource identifier addresses the collection of all instances referencing the
3211 "CIM_ManagedElement" instance in the "myns" namespace that is identified by the key list
3212 "acme,1".

3213 **B.1.12 Instance method invocation resource**

3214 A WBEM resource identifier addressing an instance method invocation resource could conform to the
3215 WBEM-resource-identifier ABNF rule (see 7.1.1) with the following additional rules:

```

3216 path-absolute = InstanceMethodInvocationPath
3217
3218 InstanceMethodInvocationPath = InstancePath "/"methodinvocation"

```

3219 Example:

- 3220 • /cimrs/namespaces/myns/classes/CIM_Example/instances/acme,1/methodinvo
- 3221 cation

3222 This resource identifier addresses the method invocation point of the "CIM_Example" instance
 3223 in the "myns" namespace that is identified by the key list "acme,1".

3224 B.1.13 Qualifier type collection resource

3225 A WBEM resource identifier addressing a qualifier type collection resource could conform to the WBEM-
 3226 resource-identifier ABNF rule (see 7.1.1) with the following additional rules:

```

3227 path-absolute = QualifierTypeCollectionPath
3228
3229 QualifierTypeCollectionPath = NamespacePath "/"qualifiers"

```

3230 Example:

- 3231 • /cimrs/namespaces/myns/qualifiers

3232 This resource identifier addresses the collection of all CIM qualifier types in the "myns"
 3233 namespace.

3234 B.1.14 Qualifier type resource

3235 A WBEM resource identifier addressing a qualifier type resource could conform to the WBEM-resource-
 3236 identifier ABNF rule (see 7.1.1) with the following additional rules:

```

3237 path-absolute = QualifierTypePath
3238
3239 QualifierTypePath = QualifierTypeCollectionPath "/" QualifierName

```

3240 Where:

- 3241 • QualifierName is the percent-encoded name of the qualifier. (See 7.1.2 for percent-encoding
 3242 rules.)

3243 Example:

- 3244 • /cimrs/namespaces/myns/qualifiers/Description

3245 This resource identifier addresses the "Description" qualifier type in the "myns" namespace.

3246 B.1.15 Instance query resource

3247 A WBEM resource identifier addressing an instance query resource could conform to the WBEM-
 3248 resource-identifier ABNF rule (see 7.1.1) with the following additional rules:

```

3249 path-absolute = InstanceQueryPath
3250
3251 InstanceQueryPath = NamespacePath "/"instancequery"

```

3252 Example:

- 3253 • /cimrs/namespaces/myns/instancequery

3254 This resource identifier addresses the instance query resource of the "myns" namespace.

3255 **B.2 Example using WBEM URIs**

3256 Because the structure of WBEM resource identifiers for non-top-level resources is not mandated in this
3257 document, WBEM URIs (defined in [DSP0207](#)) could be used as a basis. However, the addressing
3258 abilities of WBEM URIs would need to be extended to cover all resource types defined in this document.

3259 The structure of the top-level resource (that is, the namespace collection resource) still needs to conform
3260 to the structure defined in 7.2.1.

3261
3262
3263
3264
3265

ANNEX C (informative)

Change log

Version	Date	Description
1.0.0a	2010-10-11	Released as Work in Progress (of an Informational Specification)
1.0.0	2011-04-15	Released as DMTF Informational Specification

Bibliography

3266

3267 This annex contains a list of non-normative references for this document.

3268 DMTF DSP0200, *CIM Operations over HTTP 1.3*,
3269 http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf

3270 DMTF DSP0202, *CIM Query Language Specification 1.0*,
3271 http://www.dmtf.org/standards/published_documents/DSP0202_1.0.pdf

3272 DMTF DSP0207, *WBEM URI Mapping 1.0*,
3273 http://www.dmtf.org/standards/published_documents/DSP0207_1.0.pdf

3274 DMTF DSP0230, *WS-CIM Mapping Specification 1.0*,
3275 http://www.dmtf.org/standards/published_documents/DSP0230_1.0.pdf

3276 DMTF DSP1054, *Indications Profile 1.1*,
3277 http://www.dmtf.org/sites/default/files/standards/documents/DSP1054_1.1.pdf

3278 DMTF DSP-IS0202, *CIM-RS Binding to JSON 1.0.0a (Work in Progress)*,
3279 http://www.dmtf.org/standards/published_documents/DSP-IS0202_1.0.0a.pdf

3280 ITU-T X.509, *Information technology – Open Systems Interconnection – The Directory: Public-key and*
3281 *attribute certificate frameworks*,
3282 <http://www.itu.int/rec/T-REC-X.509/en>

3283 R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, PhD thesis,
3284 University of California, Irvine, 2000,
3285 <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

3286 R. Fielding, *REST APIs must be hypertext driven*, October 2008,
3287 <http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven>

3288 J. Holzer, *RESTful Web Services and JSON for WBEM Operations*, Master thesis, University of Applied
3289 Sciences, Konstanz, Germany, June 2009,
3290 <http://mond.htwg-konstanz.de/Abschlussarbeiten/Details.aspx?id=1120>

3291 A. Manes, *Rest principle: Separation of representation and resource*, March 2009,
3292 <http://apsblog.burtongroup.com/2009/03/rest-principle-separation-of-representation-and-resource.html>

3293 L. Richardson and S. Ruby, *RESTful Web Services*, May 2007, O'Reilly, ISBN 978-0-596-52926-0,
3294 <http://www.oreilly.de/catalog/9780596529260/>