



1
2
3
4
5

Document Number: DSP0210

Date: 2013-01-24

Version: 1.0.0

6 **CIM-RS Protocol**

7 **Document Type: Specification**

8 **Document Status: DMTF Standard**

9 **Document Language: en-US**

10

11 Copyright Notice

12 Copyright © 2010–2013 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

CONTENTS

34	Foreword	7
35	Introduction.....	8
36	Document conventions.....	8
37	Typographical conventions	8
38	ABNF usage conventions	8
39	Experimental material	8
40	1 Scope	9
41	2 Normative references	9
42	3 Terms and definitions	10
43	4 Symbols and abbreviated terms.....	15
44	5 Concepts	16
45	5.1 CIM-RS protocol participants.....	16
46	5.2 Model independence of CIM-RS.....	17
47	5.3 Basic kinds of resources.....	19
48	5.4 Mapping model elements to CIM-RS resources.....	19
49	5.4.1 Classes	19
50	5.4.2 Instances.....	19
51	5.4.3 Properties.....	19
52	5.4.4 Methods and operations	20
53	5.5 Two-staged mapping approach	20
54	5.6 Navigation between resources	21
55	5.7 Discovering resources in a server	24
56	5.8 REST architectural style supported by CIM-RS	25
57	6 Resource identifiers.....	26
58	6.1 CIM-RS resource identifier format	26
59	6.2 Opaqueness.....	27
60	6.3 Percent-encoding.....	27
61	6.4 Authority component.....	29
62	6.5 Query parameters.....	29
63	6.5.1 \$class (specify class name).....	31
64	6.5.2 \$continueonerror (continue on errors within paged retrieval)	31
65	6.5.3 \$expand (include target instances).....	32
66	6.5.4 \$filter (filter instances in result)	33
67	6.5.5 \$max (limit number of collection members in result)	34
68	6.5.6 \$methods (subset method links in result)	35
69	6.5.7 \$pagingtimeout (specify inactivity timeout for paged retrieval).....	36
70	6.5.8 \$properties (subset properties in result)	37
71	6.5.9 \$refer (include references to target instances)	38
72	6.6 Resource identifiers of entry point resources	39
73	7 Resources, operations and payload elements	39
74	7.1 Overview	39
75	7.2 Description conventions.....	41
76	7.2.1 Datatypes used in payload element definitions	41
77	7.2.2 Requirement levels used in payload element definitions.....	42
78	7.2.3 Requirement levels used in operation definitions	42
79	7.2.4 CIM-RS operation description format	43
80	7.3 Common behaviors for all operations	43
81	7.3.1 Content negotiation.....	43
82	7.3.2 Verifying the basis of resource modifications (EXPERIMENTAL).....	44
83	7.3.3 Caching of responses	44
84	7.3.4 Success and failure.....	44

85	7.3.5	Errors	44
86	7.3.6	ErrorResponse payload element	45
87	7.3.7	Consistency model.....	45
88	7.3.8	Paging of collections	45
89	7.4	Optional features of the CIM-RS protocol.....	47
90	7.4.1	Entity tagging feature	47
91	7.4.2	Continue on error feature.....	47
92	7.5	Instance creation resource	47
93	7.5.1	POST	48
94	7.6	Instance resource	50
95	7.6.1	Instance payload element.....	50
96	7.6.2	DELETE	51
97	7.6.3	GET	51
98	7.6.4	PUT	53
99	7.7	Reference collection resource	56
100	7.7.1	ReferenceCollection payload element	56
101	7.7.2	GET	56
102	7.8	Instance collection resource	58
103	7.8.1	InstanceCollection payload element	58
104	7.8.2	GET	59
105	7.9	Instance enumeration resource	61
106	7.9.1	GET	61
107	7.10	Method invocation resource.....	63
108	7.10.1	MethodRequest payload element	63
109	7.10.2	MethodResponse payload element	63
110	7.10.3	POST	64
111	7.11	Listener destination resource.....	66
112	7.11.1	IndicationDeliveryRequest payload element	66
113	7.11.2	POST	66
114	7.12	Server entry point resource	68
115	7.12.1	ServerEntryPoint payload element	68
116	7.12.2	GET	69
117	7.13	Listener entry point resource	71
118	7.13.1	ListenerEntryPoint payload element	71
119	7.13.2	GET	72
120	7.14	CIM-RS resources to be exposed.....	73
121	7.14.1	Resources exposed by a server	73
122	7.14.2	Resources exposed by a listener.....	74
123	7.15	Other typical WBEM protocol functionality.....	74
124	7.15.1	Server discovery	74
125	7.15.2	Discovery of server and listener entry point resources.....	74
126	7.15.3	Namespace discovery	74
127	7.15.4	Registered profile discovery	74
128	7.15.5	Schema inspection.....	74
129	7.15.6	Association traversal.....	74
130	7.15.7	Indication subscription	75
131	8	HTTP usage	75
132	8.1	General requirements	75
133	8.2	Authentication requirements	75
134	8.2.1	Operating without authentication	75
135	8.2.2	HTTP basic authentication.....	75
136	8.2.3	HTTP digest authentication	76
137	8.2.4	Other authentication mechanisms	76
138	8.3	Message encryption requirements.....	76
139	8.4	HTTP header fields	77
140	8.4.1	Accept	77

141 8.4.2 Content-Type 78

142 8.4.3 ETag (EXPERIMENTAL) 78

143 8.4.4 If-Match (EXPERIMENTAL)..... 79

144 8.4.5 X-CIMRS-Version 79

145 9 Payload representation 80

146 9.1 Internet media types 80

147 9.1.1 General 80

148 9.1.2 Media type parameters 80

149 9.2 Payload element representations 81

150 9.3 Payload representations 81

151 10 Discovery requirements..... 82

152 11 Version compatibility 82

153 11.1 HTTP protocol version compatibility 82

154 11.2 CIM-RS protocol version compatibility 82

155 11.3 CIM-RS payload representation version compatibility 83

156 12 Conformance..... 83

157 ANNEX A (normative) Common ABNF rules 84

158 ANNEX B (informative) Mapping CIM-RS to generic operations 85

159 B.1 URI composition..... 85

160 B.2 Query parameters 87

161 B.3 Server operations..... 90

162 B.4 Listener operations 95

163 ANNEX C (informative) Mapping generic operations to CIM-RS..... 97

164 C.1 Conformance 97

165 C.2 Support of optional generic operations features 97

166 C.3 Operations supported 97

167 C.4 Operations not supported 105

168 ANNEX D (informative) Examples 107

169 D.1 Navigation between resources 107

170 D.2 Paged retrieval..... 121

171 ANNEX E (informative) Change log..... 123

172 Bibliography 124

173 **Figures**

174 Figure 1 – Participants in the CIM-RS protocol..... 17

175 Figure 2 – Single model and multiple protocols 18

176 Figure 3 – Two-staged mapping approach in CIM-RS 21

177 Figure 4 – Expanding association classes to construct navigation paths..... 23

178 Figure D-1 – Class diagram for navigation examples..... 107

179 Figure D-2 – Example instance diagram for navigation to referencing association instances 109

180 Figure D-3 – Example instance diagram for navigation to associated instances 110

181 Figure D-4 – Example instance diagram for navigation to association instances across one hop..... 111

182 Figure D-5 – Example instance diagram for navigation to associated instances across two hops 113

183 Figure D-6 – Example instance diagram for navigation to associated instances across two hops (2)..... 114

184 Figure D-7 – Example instance diagram for navigation with two paths that form a subset (merge) 115

185 Figure D-8 – Example instance diagram for navigation with two paths that have a common begin 117

186 Figure D-9 – Example instance diagram for expansion of association reference 118

187 Figure D-10 – Example instance diagram for navigation starting from association..... 119

188 Figure D-11 – Example instance diagram for expansion of association reference and navigation to

189 referencing association (merge) 120

190

191 **Tables**

192	Table 1 – Query parameters in CIM-RS	30
193	Table 2 – Resource types in CIM-RS	39
194	Table 3 – CIM-RS operations.....	40
195	Table 4 – CIM-RS payload elements	40
196	Table 5 – Datatypes used in payload elements	41
197	Table 6 – Attributes of an ErrorResponse payload element	45
198	Table 7 – Operations supporting paging of collections	46
199	Table 8 – Attributes of an Instance payload element.....	50
200	Table 9 – Attributes of an ReferenceCollection payload element.....	56
201	Table 10 – Attributes of an InstanceCollection payload element.....	58
202	Table 11 – Attributes of a MethodRequest payload element.....	63
203	Table 12 – Attributes of a MethodResponse payload element	64
204	Table 13 – Attributes of an IndicationDeliveryRequest payload element	66
205	Table 14 – Attributes of a ServerEntryPoint payload element	68
206	Table 15 – Attributes of SEPNamespace payload datatype	69
207	Table 16 – Attributes of a ListenerEntryPoint payload element.....	71
208	Table 17 – Media type parameters	80
209	Table 18 – CIM-RS payload representations.....	82
210	Table B-1 – Mapping of CIM-RS query parameters to generic operations input parameters	88
211	Table C-1 – Pulled equivalents of direct instance enumeration operations.....	105
212		

213

Foreword

214 The CIM-RS Protocol (DSP0210) specification was prepared by the DMTF CIM-RS Working Group,
215 based on work of the DMTF CIM-RS Incubator.

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

218 Acknowledgments

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

- 220 • Cornelia Davis, EMC
- 221 • George Ericson, EMC
- 222 • Johannes Holzer, IBM
- 223 • Robert Kieninger, IBM
- 224 • Wojtek Kozaczynski, Microsoft
- 225 • Larry Lamers, VMware
- 226 • Andreas Maier, IBM (editor)
- 227 • Bob Tillman, EMC
- 228 • Marvin Waschke, CA Technologies

229

Introduction

230 The information in this document should be sufficient to unambiguously identify the protocol interactions
 231 that shall be supported when implementing the CIM-RS protocol. The CIM-RS protocol follows the
 232 principles of the REST architectural style for accessing modeled resources whose model conforms to the
 233 CIM metamodel defined in [DSP0004](#).

234 The target audience for this document is implementers of WBEM servers, clients, and listeners that
 235 support the CIM-RS protocol.

236 Document conventions

237 Typographical conventions

238 The following typographical conventions are used in this document:

- 239 • Document titles are marked in *italics*.
- 240 • ABNF rules and JSON text are in `monospaced font`.

241 ABNF usage conventions

242 Format definitions in this document are specified using ABNF (see [RFC5234](#)), with the following
 243 deviations and additions:

- 244 • Literal strings are to be interpreted as case-sensitive UCS characters, as opposed to the
 245 definition in [RFC5234](#) that interprets literal strings as case-insensitive US-ASCII characters.
- 246 • The hash character "#" is used to denote a comma separated list of the rule following the hash
 247 character (similar to how "*" indicates a list of the rule following it, just without separator
 248 characters). The separator comma may be surrounded by linear whitespace, empty list items
 249 (that is, comma followed by comma) get eliminated, and multiplicity modifiers are supported, as
 250 described for "#rule" in section 2.1 of [RFC2616](#).

251 The following general ABNF rules are defined:

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

253 Experimental material

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

259 The following typographical convention indicates experimental material:

260 EXPERIMENTAL

261 Experimental material appears here.

262 EXPERIMENTAL

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

265

CIM-RS Protocol

1 Scope

267 The DMTF defines requirements for interoperable communication between various clients and servers for
268 the purposes of Web Based Enterprise Management (WBEM).

269 REST architectural style was first described by Roy Fielding in chapter 5 of [Architectural Styles and the](#)
270 [Design of Network-based Software Architectures](#) and in [REST APIs must be hypertext driven](#). This style
271 generally results in simple interfaces that are easy to use and that do not impose a heavy burden on
272 client side resources.

273 This document describes the CIM-RS Protocol, which applies the principles of the REST architectural
274 style for a communications protocol between WBEM clients, servers, and listeners.

275 The DMTF base requirements for interoperable communication between WBEM clients and servers are
276 defined collectively by [DSP0004](#) and [DSP0223](#). These specifications form the basis for profiles (see
277 [DSP1001](#)) that define interfaces for specific management purposes.

278 The semantics of CIM-RS protocol operations are first described in a standalone manner and then are
279 mapped to the generic operations defined in [DSP0223](#).

280 It is a goal that a protocol adapter can be implemented on a WBEM server that enables a RESTful client
281 interface utilizing CIM-RS to access the functionality implemented on that server. It is also a goal that an
282 adapter can be written that enables WBEM clients to translate client operations into CIM-RS protocol
283 operations.

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

285 The CIM-RS protocol supports multiple resource representations; these are described in separate
286 payload representation specifications. Their use within the CIM-RS protocol is determined through HTTP
287 content negotiation. See 9.3 for a list of known payload representations and requirements for
288 implementing them.

289 Background information for CIM-RS is described in a white paper, [DSP2032](#).

2 Normative references

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

295 DMTF DSP0004, *CIM Infrastructure Specification 2.7*,
296 http://www.dmtf.org/standards/published_documents/DSP0004_2.7.pdf

297 DMTF DSP0205, *WBEM Discovery Using SLP 1.0*,
298 http://www.dmtf.org/standards/published_documents/DSP0205_1.0.pdf

299 DMTF DSP0206, *WBEM SLP Template 2.0*,
300 http://www.dmtf.org/standards/published_documents/DSP0206_2.0.txt

301 DMTF DSP0212, *Filter Query Language 1.0*,
302 http://www.dmtf.org/standards/published_documents/DSP0212_1.0.pdf

- 303 DMTF DSP0223, *Generic Operations 1.0*,
304 http://www.dmtf.org/standards/published_documents/DSP0223_1.0.pdf
- 305 DMTF DSP0211, *CIM-RS Payload Representation in JSON 1.0*,
306 http://www.dmtf.org/standards/published_documents/DSP0211_1.0.pdf
- 307 IETF RFC2246, *The TLS Protocol Version 1.0*, January 1999,
308 <http://tools.ietf.org/html/rfc2246>
- 309 IETF RFC2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999,
310 <http://tools.ietf.org/html/rfc2616>
- 311 IETF RFC2617, *HTTP Authentication: Basic and Digest Access Authentication*, June 1999,
312 <http://tools.ietf.org/html/rfc2617>
- 313 IETF RFC2818, *HTTP Over TLS*, May 2000,
314 <http://tools.ietf.org/html/rfc2818>
- 315 IETF RFC3986, *Uniform Resource Identifier (URI): Generic Syntax*, January 2005,
316 <http://tools.ietf.org/html/rfc3986>
- 317 IETF RFC4346, *The Transport Layer Security (TLS) Protocol, Version 1.1*, April 2006,
318 <http://tools.ietf.org/html/rfc4346>
- 319 IETF RFC5234, *Augmented BNF for Syntax Specifications: ABNF*, January 2008,
320 <http://tools.ietf.org/html/rfc5234>
- 321 IETF RFC5246, *The Transport Layer Security (TLS) Protocol, Version 1.2*, August 2008,
322 <http://tools.ietf.org/html/rfc5246>
- 323 ISO/IEC 10646:2003, *Information technology -- Universal Multiple-Octet Coded Character Set (UCS)*,
324 [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)
- 325 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards (2004, 5th*
326 *edition)*,
327 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse>
- 328 NIST Special Publication 800-57, Elaine Barker et al, *Recommendation for Key Management – Part 1:*
329 *General (Revised)*, March 2007,
330 http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf
- 331 NIST Special Publication 800-131A, Elaine Barker and Allen Roginsky, *Transitions: Recommendation for*
332 *Transitioning the Use of Cryptographic Algorithms and Key Lengths*, January 2011,
333 <http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf>
- 334 The Unicode Consortium, *The Unicode Standard, Version 5.2.0, Annex #15: Unicode Normalization*
335 *Forms*,
336 <http://www.unicode.org/reports/tr15/>

37 3 Terms and definitions

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

40 The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"),
41 "may", "need not" ("not required"), "can", and "cannot" in this document are to be interpreted as described
42 in [ISO/IEC Directives, Part 2](#), Annex H. The terms in parenthesis are alternatives for the preceding term,
43 for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that

344 [ISO/IEC Directives, Part 2](#), Annex H specifies additional alternatives. Occurrences of such additional
345 alternatives shall be interpreted in their normal English meaning.

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

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

351 The terms defined in [DSP0004](#) and [DSP0223](#) apply to this document. Specifically, this document uses
352 the terms "namespace", "qualifier", "qualifier type", "class", "creation class", "ordinary class",
353 "association", "indication", "instance", "property", "ordinary property", "reference", "method", "parameter",
354 and "return value" defined in [DSP0004](#).

355 The following additional terms are used in this document.

356 **3.1**

357 **CIM-RS operation**

358 an interaction in the CIM-RS protocol where a WBEM client invokes an action in a WBEM server, or a
359 WBEM server invokes an action in a WBEM listener. For a full definition, see 5.1.

360 **3.2**

361 **CIM-RS payload element**

362 a particular type of content of the entity body of the HTTP messages used by the CIM-RS protocol.
363 Payload elements are abstractly defined in this document, and concretely in CIM-RS payload
364 representation specifications. For the list of payload elements defined for the CIM-RS protocol, see Table
365 4.

366 **3.3**

367 **CIM-RS payload representation**

368 an encoding format that defines how the abstract payload elements defined in this document are encoded
369 in the entity body of the HTTP messages used by the CIM-RS protocol. This includes resource
370 representations. For more information, see clause 9.

371 **3.4**

372 **CIM-RS payload representation specification**

373 a specification that defines a CIM-RS payload representation. For more information, see clause 9.

374 **3.5**

375 **CIM-RS protocol**

376 the protocol defined in this document and related documents.

377 **3.6**

378 **CIM-RS resource**

379 an entity in a WBEM server or WBEM listener that can be referenced using a CIM-RS resource identifier
380 and thus can be the target of an HTTP method in the CIM-RS protocol. Also called "resource" in this
381 document.

- 382 **3.7**
383 **CIM-RS resource identifier**
384 a URI that is a reference to a CIM-RS resource in a WBEM server or WBEM listener, as defined in 6. Also
385 called "resource identifier" in this document.
- 386 **3.8**
387 **HTTP basic authentication**
388 a simple authentication scheme for use by HTTP and HTTPS that is based on providing credentials in
389 HTTP header fields. It is defined in [RFC2617](#).
- 390 **3.9**
391 **HTTP content negotiation**
392 a method for selecting a representation of content in an HTTP response message when there are multiple
393 representations available. It is defined in section 12 of [RFC2616](#). Its use in the CIM-RS protocol is
394 described in 7.3.1.
- 395 **3.10**
396 **HTTP digest authentication**
397 an authentication scheme for use by HTTP and HTTPS that is based on verifying shared secrets that are
398 not exchanged. It is defined in [RFC2617](#).
- 399 **3.11**
400 **HTTP entity body**
401 the payload within an HTTP message, as defined in section 7.2 of [RFC2616](#).
- 402 **3.12**
403 **HTTP entity-header field**
404 a header field that may be used in HTTP requests and HTTP response messages, specifying information
405 that applies to the data in the entity body. Also called "HTTP entity-header".
- 406 **3.13**
407 **HTTP extension-header field**
408 an entity-header field used for custom extensions to the standard set of header fields defined in
409 [RFC2616](#). Also called "HTTP extension-header".
- 410 **3.14**
411 **HTTP general-header field**
412 a header field that may be used in HTTP requests and HTTP response messages, specifying information
413 that applies to the HTTP message. Also called "HTTP general-header".
- 414 **3.15**
415 **HTTP header field**
416 a named value used in the header of HTTP messages, as defined in section 4.2 of [RFC2616](#). Also called
417 "HTTP header". The specific types of header fields are general-header field, request-header field,
418 response-header field, entity-header field, and extension-header field.
- 419 **3.16**
420 **HTTP message**
421 an interaction between an HTTP client and an HTTP server (in any direction), as defined in section 4 of
422 [RFC2616](#).
- 423 **3.17**
424 **HTTP method**
425 the type of interaction stated in HTTP requests, as defined in section 5.1.1 of [RFC2616](#).

426 3.18**427 HTTP request message**

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

430 3.19**431 HTTP request-header field**

432 a header field that may be used in HTTP requests, specifying information that applies to the HTTP
433 message. Also called "HTTP request-header".

434 3.20**435 HTTP response message**

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

438 3.21**439 HTTP response-header field**

440 a header field that may be used in HTTP response messages, specifying information that applies to the
441 HTTP message. Also called "HTTP response-header".

442 3.22**443 Internet media type**

444 a string identification for representation formats in Internet protocols. Originally defined for email
445 attachments and termed "MIME type". Because the CIM-RS protocol is based on HTTP, it uses the
446 definition of media types from section 3.7 of [RFC2616](#).

447 3.23**448 Interop namespace**

449 a role of a CIM namespace for the purpose of providing a common and well-known place for clients to
450 discover modeled entities, such as the profiles to which an implementation advertises conformance. The
451 term is also used for namespaces that assume that role. For details, see [DSP1033](#).

452 3.24**453 method invocation link**

454 the resource identifier of a (static or instance) method invocation resource (see 7.10).

455 3.25**456 model**

457 a model (including, but not limited to, the CIM Schema published by DMTF), that conforms to the CIM
458 metamodel defined in [DSP0004](#). A model may in addition conform to management profiles (see
459 [DSP1001](#)).

460 3.26**461 navigation property**

462 a property in the REST representation of an instance that is not declared in its class but is included in the
463 representation to provide for navigation to related instances. See 5.6 for details.

464 3.27**465 Normalization Form C**

466 a normalization form for UCS characters that avoids the use of combining marks where possible and that
467 allows comparing UCS character strings on a per-code-point basis. It is defined in [The Unicode Standard,](#)
468 [Annex #15](#).

- 469 **3.28**
470 **reference-typed parameter**
471 a CIM method parameter declared with a CIM datatype that is a reference to a specific class.
- 472 **3.29**
473 **reference-typed property**
474 a CIM property declared with a CIM datatype that is a reference to a specific class. See 5.4.3 for details.
475 [DSP0004](#) defines the term "reference" for such properties; this document uses the more specific term
476 "reference-typed property", instead.
- 477 **3.30**
478 **reference-qualified property**
479 a string-typed CIM property qualified with the *Reference* qualifier (see [DSP0004](#) for a definition of the
480 *Reference* qualifier, and 5.4.3 for details).
- 481 **3.31**
482 **reference property**
483 a general term for reference-typed properties and reference-qualified properties. See 5.4.3 for details.
- 484 **3.32**
485 **resource representation**
486 a representation of a resource or some aspect thereof, in some format. A particular resource may have
487 any number of representations. The format of a resource representation is identified by a media type. In
488 the CIM-RS protocol, the more general term "payload representation" is used, because not all payload
489 elements are resource representations.
- 490 **3.33**
491 **REST architectural style**
492 the architectural style described in [Architectural Styles and the Design of Network-based Software](#)
493 [Architectures](#), chapter 5, and in [REST APIs must be hypertext driven](#).
- 494 **3.34**
495 **UCS character**
496 a character from the Universal Character Set defined in [ISO/IEC 10646:2003](#). See also [DSP0004](#) for the
497 usage of UCS characters in CIM strings. An alternative term is "Unicode character".
- 498 **3.35**
499 **WBEM client**
500 the client role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see 5.1.
- 501 **3.36**
502 **WBEM listener**
503 the event listener role in the CIM-RS protocol and in other WBEM protocols.. For a full definition, see 5.1.
- 504 **3.37**
505 **WBEM server**
506 the server role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see 5.1.

507 4 Symbols and abbreviated terms

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

510 4.1

511 **ABNF**

512 Augmented Backus-Naur Form, as defined in [RFC5234](#).

513 4.2

514 **CIM**

515 Common Information Model, as defined by DMTF.

516 4.3

517 **CIM-RS**

518 **CIM RESTful Services**

519 the name of the protocol defined in this document and related documents.

520 4.4

521 **FQL**

522 Filter Query Language, as defined by DMTF.

523 4.5

524 **HTTP**

525 Hyper Text Transfer Protocol. HTTP version 1.1 is defined in [RFC2616](#). Unless otherwise noted, the term
526 HTTP is used in this document to mean both HTTP and HTTPS.

527 4.6

528 **HTTPS**

529 Hyper Text Transfer Protocol Secure, as defined in [RFC2818](#).

530 4.7

531 **IANA**

532 Internet Assigned Numbers Authority; see <http://www.iana.org>.

533 4.8

534 **JSON**

535 JavaScript Object Notation, as defined in [ECMA-262](#).

536 4.9

537 **REST**

538 Representational State Transfer, as originally and informally described in [Architectural Styles and the
539 Design of Network-based Software Architectures](#).

540 4.10

541 **SLP**

542 Server Location Protocol, as defined in [RFC2608](#).

543 4.11

544 **UCS**

545 Universal Character Set, as defined in [ISO/IEC 10646:2003](#).

- 546 **4.12**
547 **URI**
548 Uniform Resource Identifier, as defined in [RFC3986](#).
- 549 **4.13**
550 **UTF-8**
551 UCS Transformation Format 8, as defined in [ISO/IEC 10646:2003](#).
- 552 **4.14**
553 **WBEM**
554 Web Based Enterprise Management, as defined by DMTF.
- 555 **4.15**
556 **XML**
557 eXtensible Markup Language, as defined by W3C.

558 **5 Concepts**

559 This clause defines concepts of the CIM-RS protocol.

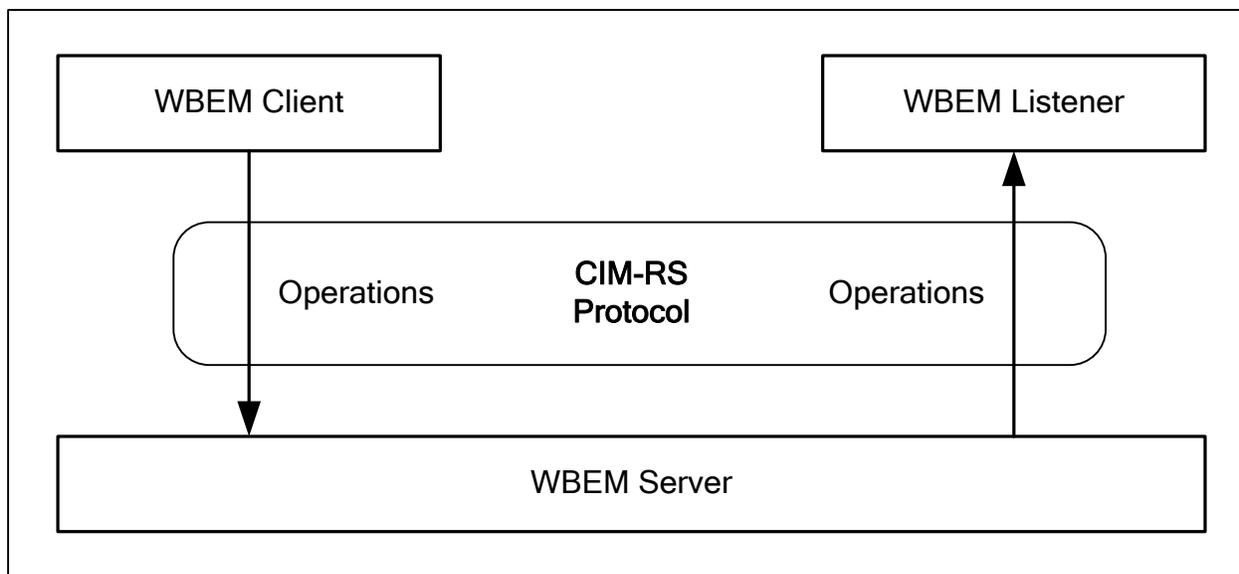
560 **5.1 CIM-RS protocol participants**

561 The participants in the CIM-RS protocol are the same as those for other WBEM protocols (for example,
562 CIM-XML): *operations* are directed from WBEM client to WBEM server, and from WBEM server to WBEM
563 listener (mainly for delivering indications, that is, event notifications). These operations are identified by
564 their HTTP method and target resource type, for example: "HTTP GET on an instance resource".

565 In this document, the terms *client*, *server*, and *listener* are used as synonyms for WBEM client, WBEM
566 server, and WBEM listener, respectively.

567 Separating the roles for client and listener in the protocol definition makes it easier to describe
568 implementations that separate these roles into different software components. Both of these roles can be
569 implemented in the same management application.

570 Figure 1 shows the participants in the CIM-RS protocol.



571
572

573 **Figure 1 – Participants in the CIM-RS protocol**

574 5.2 Model independence of CIM-RS

575 A WBEM server implements management services based on a [DSP0004](#) conformant model composed of
576 some number of modeled objects. [DSP0004](#) conformant models are defined with commonly used model
577 elements, including complex types, classes, and relationships between instances of classes.

578 The modeled objects represent entities (managed objects) in the managed environment (that is, the real
579 world). The model defines the modeled objects, their state and behavior and the relationships between
580 them. In the protocol-neutral [DSP0004](#) terminology, modeled objects are termed "instances"; in REST
581 parlance, the modeled objects are termed "resources". The CIM-RS protocol provides access to those
582 resources. The term "resource" is used in this document for anything that can be the target of an HTTP
583 method; this includes more kinds of resources than just those that represent instances.

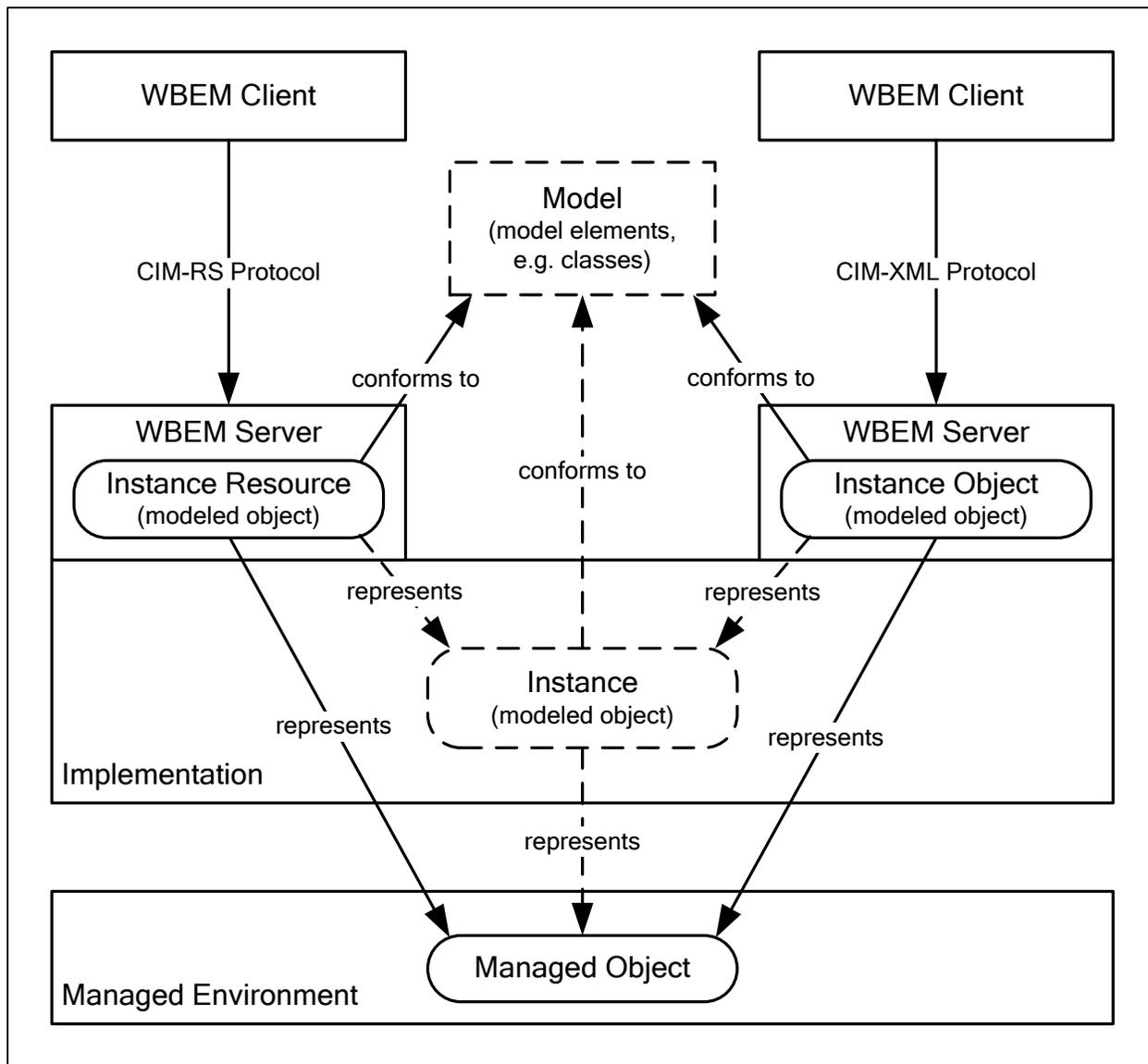
584 The CIM Schema published by DMTF is an example of a model that is conformant to [DSP0004](#), but any
585 [DSP0004](#) conformant model can be used with the CIM-RS protocol. Such other models are not required
586 to be derived from the CIM Schema published by DMTF. In this document, the term "model" is used for
587 any model that conforms to the CIM metamodel defined in [DSP0004](#), regardless of whether or not it is
588 derived from the CIM Schema. Also, in this document, the term "model" includes both schemas
589 (specifying classes) and management profiles (specifying the use of classes for specific management
590 domains).

591 The definition of the CIM-RS protocol (this document) is independent of models. CIM-RS payload
592 representations should also be designed such that their definition is independent of models. This allows
593 support for CIM-RS to be added to existing WBEM implementations at the level of protocol adapters once
594 and forever, without causing additional development efforts specific for each new model. Also, support for
595 a specific model in a WBEM server can be implemented independent of whether it is accessed with CIM-
596 RS or any other WBEM protocols (this also follows the principle of model independence). This approach
597 enables CIM-RS to provide existing WBEM infrastructures with an efficient means to support RESTful
598 clients.

599 Figure 2 shows how multiple clients interact with the same managed object using different protocols but
600 the same model. In this figure, the CIM-RS protocol and the CIM-XML protocol are shown as examples.
601 Each protocol makes protocol-specific notions of modeled objects available to its clients, but these
602 different notions all conform to the same model. The instance in the middle of the picture is a protocol-

603 neutral notion of a modeled object. Whether or not such protocol-neutral instances are materialized as
 604 run-time entities is an implementation detail; only the protocol-specific notions of modeled objects are
 605 observable by clients.

606 This document uses the term "represents" as shown in the figure: The CIM-RS protocol specific instance
 607 resource represents the managed object as much as the protocol-neutral instance does. This document
 608 also uses the verbiage that an "instance resource represents an instance", when a model-level and
 609 protocol-neutral terminology is needed.



610
611

Figure 2 – Single model and multiple protocols

612
 613 The separation of protocol and model at the specification level is beneficial for and targeted to
 614 infrastructures that also separate protocol and model (for example, CIMOM/provider-based WBEM
 615 servers, or WBEM client libraries). However, such a separation in the infrastructure is not required and
 616 CIM-RS can also be implemented in REST infrastructures without separating protocol and model.

617 **5.3 Basic kinds of resources**

618 In the CIM-RS protocol, there are three basic kinds of resources:

- 619 • **Instance resources** represent a managed object in the managed environment.
- 620 • **Collection resources** represent an ordered collection of items, such as instance resources or
621 references to instance resources.
- 622 • **Invocation resources** provide the ability to invoke operations that are outside the scope of the
623 CRUD (Create, Read, Update, Delete) operations.

624 **5.4 Mapping model elements to CIM-RS resources**

625 This subclause informally describes how the elements of a model are represented as CIM-RS resources .

626 **5.4.1 Classes**

627 Classes in a model describe what aspects of the managed objects in the managed environment show up
628 in the model; they define a modeled object.

629 There are two principal uses of classes: One describes a particular object's state and behaviors. The
630 other describes the state and behaviors of a relationship between two or more objects. These are referred
631 to as "ordinary classes" and "association classes", respectively.

632 Classes are not represented as CIM-RS resources. Instance creation, enumeration of instances by class,
633 and invocation of static methods works through global invocation resources. Static properties are
634 represented like non-static properties on the instances. These mapping decisions allow not having to
635 represent class objects as CIM-RS resources.

636 Inspection of the model, for example retrieving class definitions, is envisioned to be available in the future
637 through a schema inspection model, based solely on instance-level operations.

638 **5.4.2 Instances**

639 Addressable instances of ordinary classes and association classes are represented as CIM-RS
640 resources; these are referred to as *instance resources* (see 7.6).

641 The properties of instances are represented as properties of the instance resource.

642 Behaviors of instances are the class-defined (extrinsic) methods and certain built-in (intrinsic) operations;
643 they are represented as HTTP methods either directly on the instance resource, or on specific invocation
644 resources related to the instance resource (see 5.4.4).

645 NOTE: Instances of indication classes and embedded instances are not represented as instance resources
646 because they are not addressable. Instead, they are embedded into payload elements.

647 **5.4.3 Properties**

648 Properties of addressable instances are represented as properties of the corresponding instance
649 resources. Properties of instances that are not addressable are represented as properties of the
650 corresponding instances embedded in payload elements.

651 Static properties are represented like non-static properties: In the instance resources or embedded
652 instances. As a result, a static property defined in a class is included in all instances of the class (and has
653 the same value in all these instances).

654 The term "reference properties" in CIM-RS is used for the following two kinds of properties:

- 655 • reference-typed properties – These are reference properties in association classes that are
656 declared with a CIM datatype that is a reference to a specific class; they are the ends of
657 associations. Reference-typed properties are always scalars; there are no arrays of reference-
658 typed properties. The value of a reference-typed property references a single instance.
- 659 • reference-qualified properties – These are string-typed properties that are qualified with the
660 *Reference* qualifier. These properties can be used in ordinary classes; they are like simple
661 pointers to instances and do not constitute association ends or imply any associations.
662 Reference-qualified properties may be scalars or arrays. The value of a reference-qualified
663 scalar property and the value of an array entry of a reference-qualified array property reference
664 a single instance.

665 The values of properties (including reference properties) are represented as defined for the
666 "ElementValue" payload datatype in Table 5.

667 **5.4.4 Methods and operations**

668 Class-defined (extrinsic) methods can be defined as being static or non-static. Non-static methods that
669 are implemented are exposed via method invocation links in each instance (see 7.6). Static methods that
670 are implemented are exposed via method invocation links in the global server entry point resource (see
671 7.12). Details on method invocation links are defined in Table 5.

672 CIM-RS supports a set of built-in operations that are not class-defined. These operations are the typical
673 CRUD (Create, Read, Update, Delete) operations of REST environments; they are invoked by means of
674 HTTP methods: GET, PUT, and DELETE directly on the instance resource for reading, updating and
675 deleting, respectively (see 7.6), and POST on a global instance creation resource for creating (see 7.5).

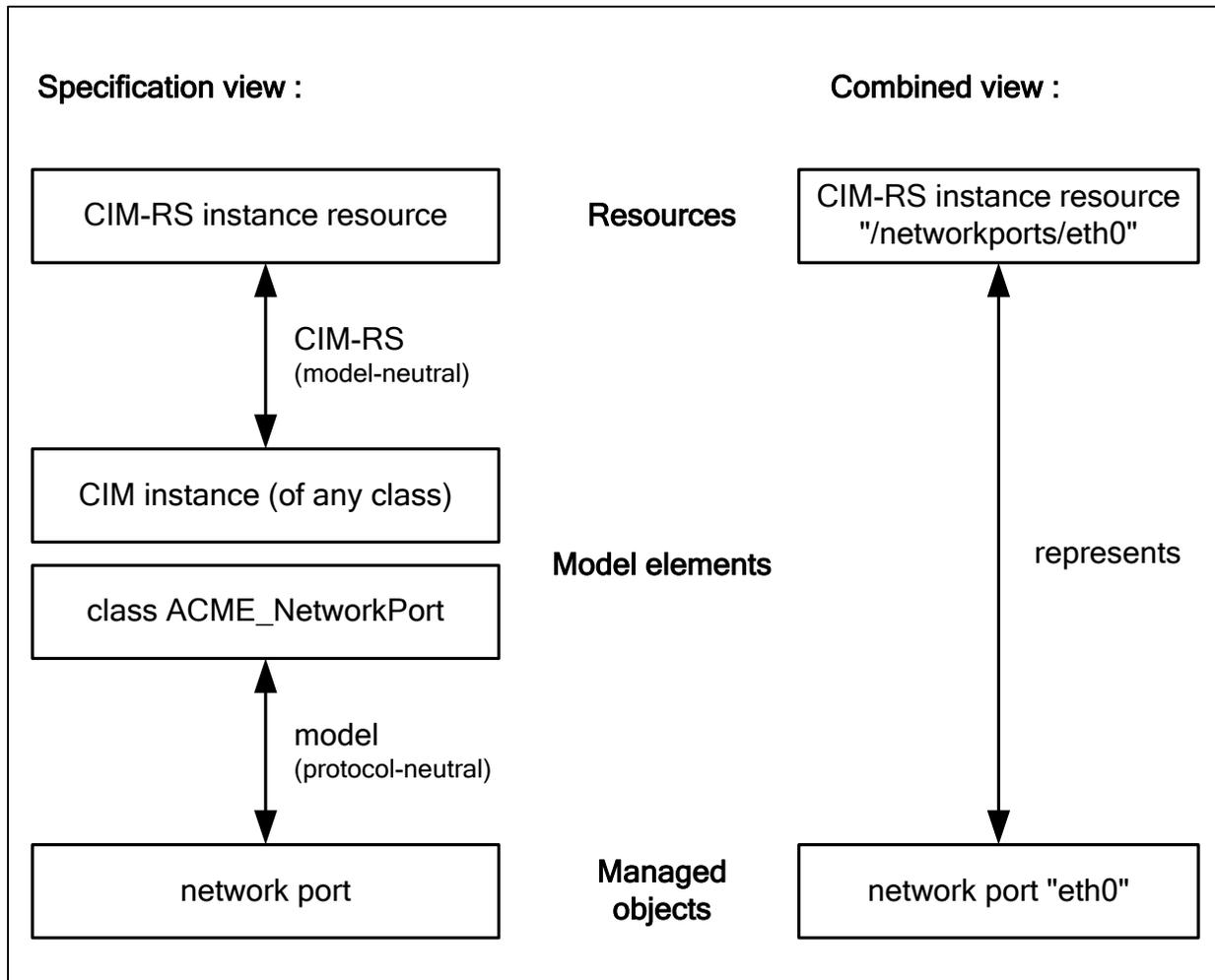
676 **5.5 Two-staged mapping approach**

677 The mapping of managed objects to CIM-RS resources uses a two-staged approach in CIM-RS, because
678 the definition of CIM-RS is model-neutral.

679 For example, let's assume that a model defines that an ACME_NetworkPort class models a managed
680 object of type "network interface". CIM-RS defines how instances of any class are represented as
681 instance resources. In combination, this describes how an instance resource of class ACME_NetworkPort
682 represents a network interface.

683 As a result, we can say that CIM-RS represents managed objects as (modeled) instance resources.

684 Figure 3 shows a pictorial representation of this two-staged mapping approach:



685

686

Figure 3 – Two-staged mapping approach in CIM-RS

687 The left side of the figure shows a specification view: The CIM-RS protocol defines how instances of any
 688 class are represented as CIM-RS instance resources. The model defines how managed objects are
 689 modeled as classes.

690 The combined view suggests that the managed objects are represented as REST instance resources.

691 **5.6 Navigation between resources**

692 Clients can navigate between resources in any of these ways:

- 693 • dereferencing resource identifiers already known, by issuing an HTTP GET on the resource
 694 identifier (see 7.6.3)
- 695 • expanding existing reference properties (typed or qualified) to the instances they reference via
 696 an \$expand (see 6.5.3) query parameter
- 697 • including *navigation properties* via an \$expand or \$refer (see 6.5.9) query parameter

698 Because of the simplicity of the first way listed above, this subclause covers only the second and third
 699 way in its remainder.

700 Navigation properties are not declared in the class of an instance, but are caused to be included in the
 701 representation of an instance as a result of specifying the \$expand or \$refer query parameters when
 702 retrieving an instance resource or instance collection resource.

703 The values of the \$expand and \$refer query parameters are lists of navigation paths.

704 A navigation path identifies the instances that are the target of the navigation, as a path across navigation
 705 hops. Each navigation hop identifies a set of instances based on the set of instances at the previous hop.

706 If a navigation path identifies an existing reference, its value gets expanded to the referenced instances
 707 when used in \$expand. Such navigation paths can also be used with \$refer; the effect is a no-op unless
 708 class-based filtering is specified (see 6.5.9).

709 If a navigation path does not identify an existing reference or an already included navigation property, a
 710 navigation property is included.

711 The value of navigation properties included due to the usage of \$refer is a reference or collection of
 712 references to these identified target instances, while the value of navigation properties included due to the
 713 usage of \$expand is the identified target instance or collection of target instances. For more details on the
 714 values of navigation properties and on the query parameter syntax, see the descriptions of \$expand (see
 715 6.5.3) and \$refer (see 6.5.9).

716 Navigation paths shall conform to the ABNF rule `nav-path`:

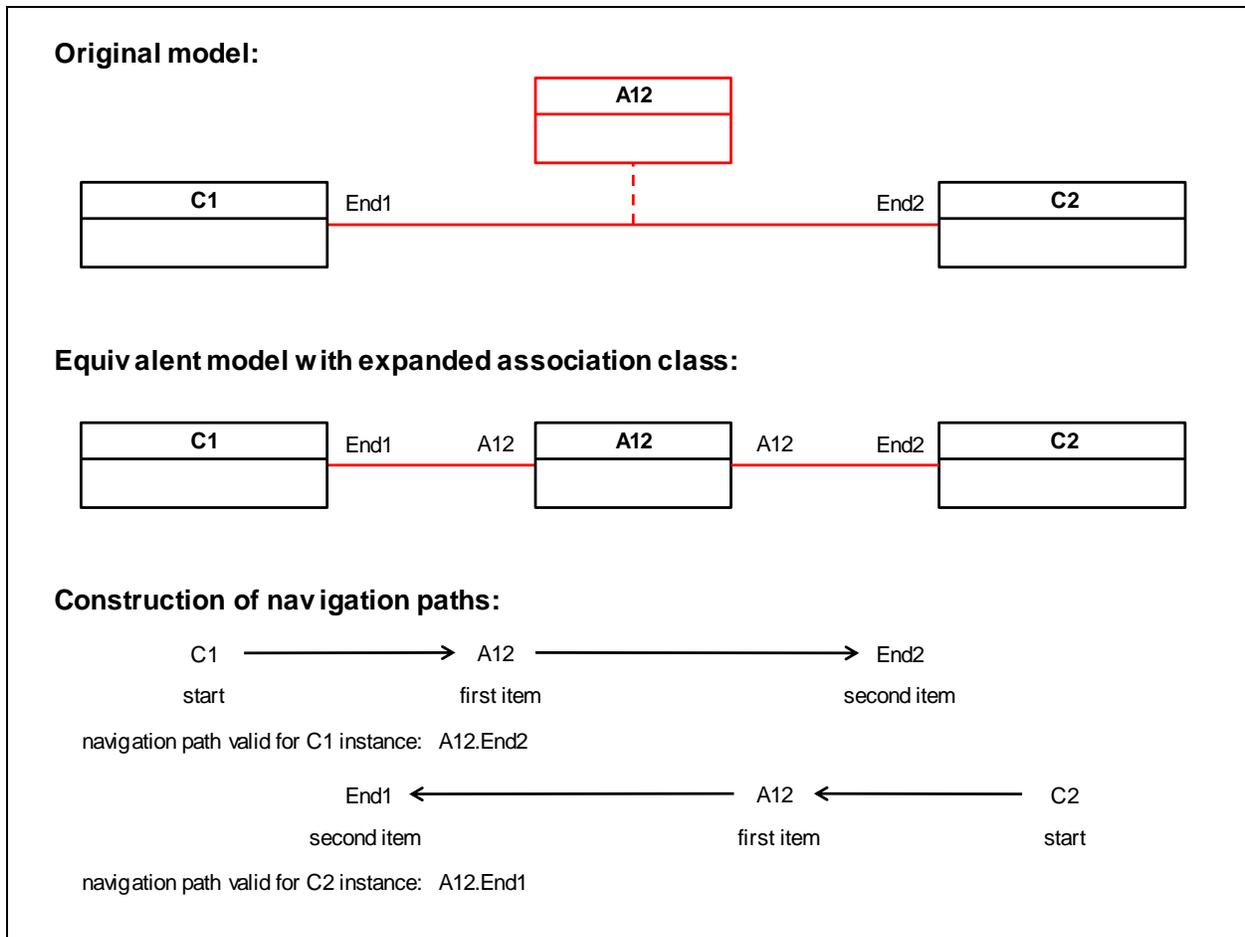
```
717 nav-path = nav-hop *( "." nav-hop )
718
719 nav-hop = nav-filter ( embedded-path ref-name / assoc-class-name )
720
721 embedded-path = *( prop-name "." )
722
723 nav-filter = ( "[" filter-class-name "]" )
```

724 Where:

- 725 • `nav-hop` identifies a set of instances at the current hop, based on the instances at the previous
 726 hop, as follows:
 - 727 – If `ref-name` is specified in `nav-hop`, `ref-name` shall either be the name of an existing
 728 (typed or qualified) reference exposed by the instances at the current hop, or the name of a
 729 navigation property of type reference that was included into the instances at the current
 730 hop on behalf of some other navigation path.
 731 `nav-hop` then identifies the instance or instances referenced by `ref-name`.
 - 732 – If `assoc-class-name` is specified in `nav-hop`, `assoc-class-name` shall be the name
 733 of an association class that references one of the classes (including subclasses) of the
 734 instances at the current hop.
 735 `nav-hop` then identifies the instance or instances referenced by `ref-name` in `filtered-`
 736 `ref`.
- 737 • `nav-filter`, when specified at a hop, filters the set of instances at that hop to be only
 738 instances of class `filter-class-name` (including instances of its subclasses). Note that such
 739 filtering can be used with both `ref-name` and `assoc-class-name`.
- 740 • `embedded-path` specifies a path through embedded instances, in case the reference is in an
 741 embedded instance. `embedded-path` starts with the property that is visible in the set of
 742 instances at the current hop (the outermost embedded instance) and ends with the property
 743 whose value is the embedded instance that has the reference as a member (the innermost
 744 embedded instance).

745 Examples of retrievals using the \$expand and \$refer query parameters are shown in D.1.

746 One way this approach for constructing navigation paths can easily be understood and remembered, is to
 747 consider that an equivalent model for an association class is to expand the association class so that it
 748 becomes a non-association class and its references become associations. This is shown in Figure 4.



749
750

751 **Figure 4 – Expanding association classes to construct navigation paths**

752 In the equivalent model, the ends of the two new associations that are directed back to the former
 753 association class get the name of the association class. A navigation path is now simply the set of far
 754 ends in navigation direction, from some starting point. This is shown in the figure for the starting point C1,
 755 where the navigation path for navigating to the C2 instances is "A12.End2", and for the starting point C2,
 756 where the navigation path for navigating to the C1 instances is "A12.End1".

757 Navigation paths identify their target instances as follows:

- 758 • Navigation paths that end with a reference name (filtered or not) identify the instance(s)
759 referenced by that ending reference
- 760 • Navigation paths that end with an association class name identify these association instances

761 For each navigation path in the \$expand and \$refer query parameters, a navigation property is included in
 762 the retrieved instance representations, unless a reference property (typed or qualified) with that name
 763 already exists. If two or more navigation paths can be merged, only one navigation property is included
 764 that has the merged name and value, as described in the following paragraphs.

765 For the purpose of merging of navigation paths, the set of navigation paths in the \$expand and \$refer
766 query parameters is treated as one single combined set.

767 Two navigation paths can be merged if the first navigation path is a subset of the second navigation path,
768 and the first navigation path was used with \$expand. Note that all navigation paths used in a particular
769 instance retrieval have the same starting point (the instance being retrieved).

770 The value of the merged navigation property is determined by identifying all elements (association
771 instances or references) in the value of the (expanded) property that would result from the first navigation
772 path alone, that are the starting points for the remainder of the second navigation path (that is, the
773 remaining string in the second navigation path after removing the portion that matches the first navigation
774 path), and by processing that remainder as a normal navigation path with the identified starting points.
775 Note that this can lead to both, expanding existing references, or including navigation properties.

776 The resulting merged property is considered to be included by \$expand, for the purpose of applying the
777 merge rule repeatedly in cases where more than two navigation properties are merged. The repeated
778 merging of two navigation properties shall be performed in the order from the shortest to the longest
779 navigation path, regardless of the order in which they were specified in the \$expand and \$refer query
780 parameters.

781 The name of a navigation property is the navigation path string without any filter classes, or the subset
782 thereof that is a valid navigation path for the navigation property given the position of the navigation
783 property in the represented instance. See D.1 for examples on these names.

784 The values of navigation properties depend on whether \$expand or \$refer was used to include them; for
785 details see 6.5.3 and 6.5.9.

786 **5.7 Discovering resources in a server**

787 This subclause provides an overview on how a client would go about discovering resources in a server,
788 using the CIM-RS protocol.

789 DMTF defines the use of SLP based discovery using the information in the *DMTF WBEM SLP Template*
790 ([DSP0206](#)). Clients can discover servers using this means (see clause 10). However, as with any WBEM
791 protocol, CIM-RS can be used without depending SLP, as long as the server is known by some means.

792 CIM-RS defines a well-known server entry point resource that may be used as a starting point for
793 discovery. Given a server URL, the client may retrieve the server entry point resource of the server using
794 an HTTP GET (see 7.12.2), using a resource identifier constructed using the well-known path component
795 of the server entry point resource (see 7.12).

796 The server entry point resource (and the listener entry point resource) are the only resources with a well-
797 known path component in their resource identifiers. Any other resource identifiers in CIM-RS are opaque
798 to clients.

799 Given a starting resource, the functionality of CIM-RS enables a client to navigate to all related resources.
800 The DMTF standard way of discovering implemented models and their entry points is described in the
801 *DMTF Profile Registration Profile* ([DSP1033](#)). The server entry point provides sufficient information for a
802 client to then utilize that standard.

803 Using the [DSP1033](#) standard, a client would start this discovery by enumerating all instances of class
804 CIM_RegisteredProfile in the Interop namespace using an HTTP GET (see 7.9.1) on the instance
805 enumeration resource. For details and how to continue from there, see [DSP1033](#). Further instances are
806 discovered either by enumerating them by class, using the instance enumeration resource (see 7.9), or
807 by traversing relationships, starting with already known instances (see 5.6).

808 5.8 REST architectural style supported by CIM-RS

809 CIM-RS follows most of the principles and constraints of the REST architectural style described by Roy
810 Fielding in chapter 5 of [Architectural Styles and the Design of Network-based Software Architectures](#) and
811 in [REST APIs must be hypertext driven](#). Any deviations from these principles and constraints are
812 described in this subclause.

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

814 • **Client-Server:** The participants in CIM-RS have a client-server relationship between a WBEM
815 client and a WBEM server. For indication delivery, there is another client-server relationship in
816 the opposite direction: The WBEM server acting as a client operates against a WBEM listener
817 acting as a server. This constraint is fully satisfied.

818 • **Stateless:** Interactions in CIM-RS are self-describing and stateless in that the WBEM server or
819 the WBEM listener do not maintain any session state. This constraint is fully satisfied.

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

828 • **Cache:** The HTTP methods used by CIM-RS are used as defined in [RFC2616](#). As a result, they
829 are cacheable as defined in [RFC2616](#). This constraint is fully satisfied.

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

831 • **Uniform interface:** The main resources represented in CIM-RS are instances or collections
832 thereof, representing modeled objects in the managed environment. CIM-RS defines a uniform
833 interface for creating, deleting, retrieving, replacing, and modifying these resources and thus the
834 represented objects, based on HTTP methods. The resource identifiers used in that interface
835 are uniformly structured. This constraint is satisfied, with the following deviation:

836 Methods can be invoked in CIM-RS through the use of HTTP POST. This may be seen as a
837 deviation from the REST architectural style, which suggests that any "method" be represented
838 as a modification of a resource. However, DMTF experience with a REST like modeling style
839 has shown that avoiding the use of methods is not always possible or convenient. For this
840 reason CIM-RS supports invocation of methods.

841 • **Layered system:** Layering is inherent to information models that represent the objects of a
842 managed environment, because clients only see the modeled representations and are not
843 exposed to the actual objects. CIM-RS defines the protocol and payload representations such
844 that it works with any model, and thus is well suited for implementations that implement a model
845 of the managed environment independently of protocols, and one or more protocols
846 independently of the model. CIM-RS works with HTTP intermediaries (for example, caches and
847 proxy servers). This constraint is fully satisfied.

848 • **Code-On-Demand:** CIM-RS does not directly support exchanging program code between the
849 protocol participants. This optional constraint is not satisfied.

850 NOTE CIM-RS support of methods enables a model to add support for exchanging program code if that
851 functionality is desired.

852 In CIM-RS, resources are addressed through resource identifiers that are URIs. The REST architectural
853 style recommends that all addressing information for a resource is in the resource identifier (and not, for
854 example, in the HTTP header). In addition, it recommends that resource identifiers are opaque to clients
855 and clients should not be required to understand the structure of resource identifiers or be required to
856 assemble any resource identifiers. CIM-RS follows the recommendations that all addressing information
857 for a resource is in the resource identifier and on opaqueness and non-assembly of the resource
858 identifier.

859 The REST architectural style promotes late binding between the abstracted resource that is addressed
860 through a resource identifier and the resource representation that is chosen in the interaction between
861 client and server. CIM-RS follows this by supporting multiple types of resource representations that are
862 chosen through HTTP content negotiation. (For details, see 7.3.1.)

863 CIM-RS supports retrieval of a subset of the properties of instances. The properties to be included in the
864 result are selected through query parameters in the resource identifier URI. Since the query component of
865 a URI is part of what identifies the resource (see [RFC3986](#)), that renders these subsetted instances to be
866 separate resources (that is, separate from the resource representing the instance with all properties),
867 following the principles of the REST architectural style.

868 The only resource identifier a WBEM client needs to have when starting to interact with a WBEM server is
869 the resource identifier of the server entry point resource of the WBEM server (see 6.6). From that point
870 on, CIM-RS operations allow discovery of the resource identifiers of any further resources, based on
871 previously returned resources.

872 This applies similarly to interactions with WBEM listeners: The only resource identifier a WBEM server
873 needs to have when starting to interact with a WBEM listener is the resource identifier of the listener entry
874 point resource of the listener (see 6.6).

875 **6 Resource identifiers**

876 Resources of the types defined in clause 7 are all accessible through the CIM-RS protocol and can be
877 addressed using a CIM-RS resource identifier. A CIM-RS resource identifier is a URI that provides a
878 means of locating the resource by specifying an access mechanism through HTTP or HTTPS. In this
879 document, the term "resource identifier" is used as a synonym for the term "CIM-RS resource identifier".

880 Usages of the resource identifier URI in the HTTP header are defined in [RFC2616](#) and [RFC2818](#). In the
881 protocol payload, resource identifiers are values of type URI (see Table 5), using the format defined in
882 6.1.

883 **6.1 CIM-RS resource identifier format**

884 This subclause defines the format of CIM-RS resource identifiers.

885 CIM-RS resource identifiers are URIs that conform to the ABNF rule `cimrs-uri`:

```
886 cimrs-uri = [ "//" authority ] path-absolute [ "?" query ]
```

887 Where:

- 888 • `authority` is defined in [RFC3986](#) and shall in addition conform to the definitions in 6.4
- 889 • `path-absolute` is defined in [RFC3986](#)
- 890 • `query` is defined in [RFC3986](#) and shall in addition conform to the definitions in 6.5

891 This format conforms to but restricts ABNF rule `URI-reference` defined in [RFC3986](#).

892 The base URI for CIM-RS resource identifiers referencing resources in a server or listener is the absolute
893 URI of its server entry point resource (see 7.12) or listener entry point resource (see 7.13), respectively.

894 The authority component in CIM-RS resource identifiers shall be present if the resource is located on a
895 different host than the host of the current HTTP communication. It should not be present if the resource is
896 located on the host of the current HTTP communication (this avoids transformations of the authority
897 component in HTTP proxies).

898 The use of fragments is not permitted in CIM-RS resource identifiers because resource identifiers serve
899 the purpose of identifying resources, and fragments are not part of the resource identification (see
900 [RFC3986](#)).

901 The scheme component (see [RFC3986](#)) is not permitted in CIM-RS resource identifiers because they are
902 intended to be independent of the access protocol (HTTP or HTTPS).

903 6.2 Opaqueness

904 In interactions between clients and servers, resource identifiers referencing resources in the server are
905 under the control of the server implementation and are opaque to clients, with the exceptions stated in
906 this subclause. Opaqueness to clients means that clients should not parse, construct or modify any such
907 resource identifiers.

908 For these interactions, the exceptions from client-opaqueness are:

- 909 • Construction of the resource identifier for the server entry point resource
- 910 • Parsing, adding, removing or modifying any query parameters in the resource identifier
- 911 • Normalizing the resource identifier, as described in [RFC3986](#) (for example, removing ".." and "."
912 segments)

913 In interactions between clients and WBEM listeners, resource identifiers referencing resources in the
914 listener are under the control of the listener implementation and are opaque to servers, with the
915 exceptions stated in this subclause. Opaqueness to servers means that servers should not parse,
916 construct or modify any such resource identifiers.

917 For these interactions, the exceptions from server-opaqueness are:

- 918 • Construction of the resource identifier for the listener entry point resource. That resource
919 identifier is typically constructed by clients and passed to the server as part of client-created
920 listener destination objects
- 921 • Parsing, adding, removing or modifying any query parameters in the resource identifier
- 922 • Normalizing the resource identifier, as described in [RFC3986](#) (for example, removing ".." and "."
923 segments)

924 6.3 Percent-encoding

925 This subclause defines how the percent-encoding rules defined in [RFC3986](#) are applied to resource
926 identifiers.

927 [RFC3986](#) defines percent-encoding for URIs in its section 2.1, resulting in the following (equivalent) rules:

- 928 • *Unreserved* characters (that is, the characters in ABNF rule `unreserved` defined in [RFC3986](#))
929 should not be percent-encoded. If they are percent-encoded, consumers of the resource
930 identifier shall tolerate that.
- 931 • The percent-encoding of *reserved* characters (that is, the characters in ABNF rule `reserved`
932 defined in [RFC3986](#)) depends on the specific query parameter and whether a character is

- 933 considered delimiter or data in that query parameter, or sometimes even within portions of the
934 query parameter.
- 935 Reserved characters that are considered delimiters shall not be percent-encoded.
- 936 Reserved characters that are considered data shall be percent-encoded.
- 937 The definitions of the query parameters in 6.5 defines which of the reserved characters are
938 considered delimiters or data, for purposes of percent-encoding.
- 939 • Any other characters (that is, outside of the ABNF rules *reserved* and *unreserved* defined in
940 [RFC3986](#)) shall be percent-encoded.
- 941 Consumers of resource identifiers shall support any percent-encoding within the resource identifier that is
942 permissible according to the rules in this subclause.
- 943 [RFC3986](#) defines percent-encoding on the basis of data octets, but it does not define how characters are
944 encoded as data octets. Because element names, namespace names, and key values may contain UCS
945 characters outside of the US-ASCII character set, this document defines the percent-encoding to be used
946 in resource identifiers as follows.
- 947 Any UCS character that is being percent-encoded in resource identifiers shall be processed by first
948 normalizing the UCS character using Normalization Form C (defined in [The Unicode Standard, Annex
949 #15](#)), then encoding it to data octets using UTF-8, and finally percent-encoding the resulting data octets
950 as defined in section 2.1 of [RFC3986](#). The requirement to use a specific Unicode normalization form and
951 a specific Unicode encoding (that is, UTF-8) ensures that the resulting string can be compared octet-wise
952 without having to apply UCS character semantics.
- 953 If values with CIM datatypes need to be represented in resource identifiers, the datatype-specific string
954 representations defined in [DSP0004](#) should be used.
- 955 The following examples use the minimally needed percent-encodings:
- 956 • The namespace name "root/cimv2" becomes "root%2Fcimv2" in a resource identifier, because
957 the slash character (/) is a reserved character in resource identifiers and we assume that the
958 usage of the namespace name has defined that an occurrence of a slash in a namespace name
959 is considered data.
- 960 • The class name "ACME_LogicalDevice" remains unchanged in a resource identifier, because it
961 contains only unreserved characters.
- 962 • The (German) key property value "ÄnderungsRate" becomes "%C3%84%0AnderungsRate" in a
963 resource identifier, because C3 84 0A are the data octets of the UTF-8 encoding of the UCS
964 character U+00C4, which represents "Ä" (A umlaut) in normalized form. Note that usage of the
965 UCS character sequence U+0061 U+0308 which also represents "Ä" (using the base character
966 "A" and the combining diacritical mark ¨) is not permitted due to the requirement to use
967 Normalization Form C.
- 968 • The string typed value "a \"brown\" bag\n" (represented using backslash escape sequences as
969 defined for string literals in MOF) becomes "a%20%22brown%22%20bag%0A" in a resource
970 identifier, because the characters blank (U+0020), newline (U+000A), and double quote
971 (U+0022) are not allowed in resource identifiers and therefore need to be percent-encoded.
- 972 • The sint8 typed value -42 becomes the string "-42" in a resource identifier, because that is the
973 string representation of an sint8 typed value defined in [DSP0004](#), and because "-" is an
974 unreserved character.

975 6.4 Authority component

976 WBEM clients, servers, and listeners shall adhere to the following additional rules regarding the value of
977 ABNF rule `authority` defined in 6.1:

- 978 • The `userinfo` component within `authority` shall not be specified because of security issues
979 with specifying an unencrypted password
- 980 • The `host` component within `authority` shall be the IP (V4 or V6) address of the server, or a
981 DNS-resolvable host name for that IP address (including "localhost")
- 982 • If the `port` component within `authority` is not specified, the port number shall default to the
983 standard port numbers for HTTP and HTTPS:
 - 984 – port number 80 when using HTTP
 - 985 – port number 443 when using HTTPS

986 If the authority component is omitted in values of type URI (see Table 5) in a request or response
987 payload, it shall default to the authority used for that operation (that is, to the value of the Host request-
988 header).

989 6.5 Query parameters

990 This subclause defines the query component of resource identifiers, and applies in addition to the
991 definition in [RFC3986](#), section 3.4.

992 The format of the query component is defined by the following ABNF rule:

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

994 Where:

- 995 • `query-parameter` is a query parameter as defined in the subclauses of this subclause
- 996 • The reserved character "&" in the literals of this ABNF rule shall be considered a delimiter for
997 purposes of percent-encoding (see 6.3)

998 Example:

- 999 • `/cimrs/networkports?$filter=Name='eth0'&properties=Name,Description`

1000 This resource identifier specifies the query parameters `$filter` with a value of `Name='eth0'` and
1001 `$properties` with a value of `Name,Description`

- 1002 • `/cimrs/networkports?$filter=Description='a%26b'`

1003 This resource identifier specifies the query parameter `$filter` with a value of
1004 `Description='a&b'`, percent-encoding the ampersand character since it is considered a
1005 delimiter in the query parameter

1006 Query parameters of resource identifiers (that is, both name and value) are case sensitive, as defined in
1007 [RFC3986](#), section 6.2.2.1, unless defined otherwise in this subclause. The query parameters defined in
1008 the subclauses of this subclause define in some cases that the values of query parameters are to be
1009 treated case insensitively. In such cases, two resource identifiers that differ only in the lexical case of
1010 query parameters address the same resource, even though the resource identifiers do not match
1011 according to the rules defined in [RFC3986](#). It is recommended that producers of resource identifiers
1012 preserve the lexical case in such case insensitive cases, in order to optimize caching based on resource
1013 identifiers. For example, if a property is named "ErrorRate", its use in the `$properties` query parameter
1014 should be `properties=ErrorRate`, preserving its lexical case.

1015 Query parameters whose syntax supports the specification of comma-separated lists of items may be
 1016 repeated; the effective list of items is the concatenation of all those lists. Any other query parameters shall
 1017 not be repeated (unless specified otherwise in the description of the query parameter); if such query
 1018 parameters are repeated in a resource identifier, the consumer of that resource identifier shall fail the
 1019 operation with HTTP status code 400 "Bad Request". The description of each query parameter will detail
 1020 whether it permits repetition.

1021 NOTE: [RFC3986](#) does not detail how the `query` ABNF rule is broken into query parameters, and thus does not
 1022 address the topic of query parameter repetition.

1023 The order and repetition of query parameters specified in resource identifiers does not matter for
 1024 purposes of identifying the resource and for the semantic of the query parameters. As a consequence,
 1025 resource identifiers need to be normalized before a simple string comparison can be used to determine
 1026 resource identity.

1027 Some query parameters are constrained to be specified only on certain resource identifiers, as defined in
 1028 the subclauses of this subclause. WBEM servers and listeners shall reject operations against resource
 1029 identifiers that do not conform to these constraints.

1030 This subclause defines the `query-parameter` rule by using ABNF incremental alternatives (that is, the
 1031 `=/` construct), based on the initially empty rule:

1032

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

1033 Table 1 lists the query parameters that shall be supported, subject to the usage constraints defined in this
 1034 document:

1035 **Table 1 – Query parameters in CIM-RS**

Query Parameter	Purpose	Description
\$class	specify class name	see 6.5.1
\$continueonerror	continue on errors within paged retrieval	see 6.5.2
\$expand	include target instances	see 6.5.3
\$filter	filter instances in result	see 6.5.4
\$max	limit number of instances in result	see 6.5.5
\$methods	subset method links in result	see 6.5.6
\$pagingtimeout	specify inactivity timeout for paged retrieval	see 6.5.7
\$properties	subset properties in result	see 6.5.8
\$refer	include references to target instances	see 6.5.9

1036 Additional implementation-defined query parameters are not permitted in CIM-RS. Note that servers (and
 1037 listeners) can use the path component of a resource identifier to include any implementation-defined
 1038 information (as long as it is opaque to the receivers).

1039 In order to prepare for query parameters to be added in future versions of this document, clients, servers
 1040 and listeners shall tolerate and ignore any query parameters not listed in Table 1. As a result, two
 1041 resource identifiers that differ only in the presence of a query parameter not listed in Table 1 address the
 1042 same resource.

1043 **6.5.1 \$class (specify class name)**

1044 The \$class query parameter is used to specify a class name for the HTTP PUT method on instance
1045 enumeration resources (see 7.9.1) or the HTTP POST method on instance creation resources (see
1046 7.5.1).

1047 The format of this query parameter is defined by the following ABNF:

```
1048 query-parameter =/ class-query-parm
1049
1050 class-query-parm = "$class=" class-name
```

1051 Where:

- 1052 • The reserved characters "\$" and "=" in the literals of these ABNF rules shall be considered
1053 delimiters for purposes of percent-encoding (see 6.3)
- 1054 • `class-name` is the name of the class (including schema prefix). Note that CIM class names do
1055 not contain reserved characters (see 6.3 and [DSP0004](#))

1056 The \$class query parameter shall not be repeated in a resource identifier.

1057 Examples:

```
1058 $class=ACME_ComputerSystem
1059
1060 specifies class name ACME_Computersystem
```

1060 **6.5.2 \$continueonerror (continue on errors within paged retrieval)**

1061 The \$continueonerror query parameter specifies whether or not the server continues paged retrieval
1062 sequences in case of errors (instead of closing them). For details about paged retrieval, see 7.3.8.

1063 The format of this query parameter is defined by the following ABNF:

```
1064 query-parameter =/ continueonerror-query-parm
1065
1066 continueonerror-query-parm = "$continueonerror" [ "=" ( "true" / "false" ) ]
```

1067 Where:

- 1068 • The reserved characters "\$" and "=" in the literals of these ABNF rules shall be considered
1069 delimiters for purposes of percent-encoding (see 6.3)

1070 Note that the values "true" and "false" are treated case sensitively, as defined in 6.3

1071 The \$continueonerror query parameter shall not be repeated in a resource identifier.

1072 Omitting the \$continueonerror query parameter or specifying it with a value of "false" shall cause the
1073 server to close paged retrieval sequences in case of errors.

1074 Specifying the \$continueonerror query parameter without a value or with a value of "true" shall cause the
1075 server to continue paged retrieval sequences in case of errors.

1076 Examples:

```
1077 (not specified)
1078 $continueonerror=false
```

1079 The server closes paged retrieval sequences in case of errors

1080 \$continueonerror
 1081 \$continueonerror=true

1082 The server continues paged retrieval sequences in case of errors

1083 **6.5.3 \$expand (include target instances)**

1084 The \$expand query parameter may be used on operations that retrieve instances or instance collections
 1085 and specifies a list of navigation paths. For details on navigation paths and the resulting navigation
 1086 properties, see 5.6.

1087 The value of navigation properties included as a result of using the \$expand query parameter shall be an
 1088 instance collection whose members are the target instances identified by the navigation path. That
 1089 instance collection shall be represented as an InstanceCollection payload element (see 7.8.1) and shall
 1090 be subject to paged retrieval (see 7.3.8).

1091 The value of existing references expanded as a result of using the \$expand query parameter depends on
 1092 the navigation path, as follows. Note that the navigation path may contain more than one hop:

- 1093 • if each hop on the navigation path is a scalar reference (typed or qualified), the value of the
 1094 expanded reference shall be the target instance identified by the navigation path. That instance
 1095 shall be represented as an Instance payload element (see 7.6.1).
- 1096 • otherwise, the value of the expanded reference shall be an instance collection whose members
 1097 are the target instances identified by the navigation path. That instance collection shall be
 1098 represented as an InstanceCollection payload element (see 7.8.1) and shall be subject to paged
 1099 retrieval (see 7.3.8).

1100 The format of the \$expand query parameter is defined by the following ABNF:

```
1101  query-parameter =/ expand-query-parm
1102
1103  expand-query-parm = "$expand=" [ expand-list ]
1104
1105  expand-list = nav-path *( "," nav-path )
```

1106 Where:

- 1107 • The reserved characters "\$", "=", and "," in the literals of these ABNF rules shall be considered
 1108 delimiters for purposes of percent-encoding (see 6.3)
- 1109 • nav-path is a navigation path identifying the target instances, as defined in 5.6; any reserved
 1110 characters in the navigation path (that is, "[" and "]") shall be considered delimiters for purposes
 1111 of percent-encoding (see 6.3). Note that the character "." in the navigation path is an
 1112 unreserved character.

1113 The \$expand query parameter may be repeated in a resource identifier, see 6.5. If repeated, the effective
 1114 expand list shall be the combined expand list of all occurrences of the \$expand query parameter.

1115 Duplicate or invalid navigation path strings in the set of all navigation paths specified for the \$expand or
 1116 \$refer query parameters shall cause the operation to fail with HTTP status code 400 "Bad Request".

1117 Examples:

1118 (not specified)
 1119 \$expand=

1120 no navigation paths have been specified; no navigation properties will be included and no
 1121 expansion of reference properties will take place

1122 \$expand=ACME_SystemDevice.PartComponent

1123 include a navigation property named "ACME_SystemDevice.PartComponent" in each retrieved
1124 instance (assuming it is valid for the retrieved instance)

1125 \$expand=Volumes

1126 expand the reference-qualified property array named "Volumes", to an instance collection of the
1127 referenced instances.

1128 For more examples, see D.1.

1129 **6.5.4 \$filter (filter instances in result)**

1130 The \$filter query parameter acts as a restricting filter on the set of instances included in an instance
1131 collection.

1132 In this version of CIM-RS, the only query language supported for the \$filter query parameter is the DMTF
1133 *Filter Query Language* (FQL) defined in [DSP0212](#).

1134 The format of this query parameter is defined by the following ABNF:

```
1135 query-parameter =/ filter-query-parm
1136
1137 filter-query-parm = "$filter=" [ filter-query ]
```

1138 Where:

- 1139 • The reserved characters "\$" and "=" in the literals of these ABNF rules shall be considered
1140 delimiters for purposes of percent-encoding (see 6.3)
- 1141 • `filter-query` is a filter query string that shall conform to the format of an FQL query string; if
1142 it evaluates to true for an instance then the instance is included, otherwise, it is not included.

1143 Any reserved characters that occur in literals of the FQL query string shall be considered data
1144 for purposes of percent-encoding.

1145 Any reserved characters that occur elsewhere in the FQL query string shall be considered
1146 delimiters for purposes of percent-encoding (see 6.3).

1147 The \$filter query parameter may be repeated in a resource identifier, see 6.5. Multiple occurrences of the
1148 \$filter query parameter shall be combined by using logical AND on the filter query of each single
1149 parameter value.

1150 The \$filter query parameter may be specified only in resource identifiers of instance collection resources.

1151 Navigation properties cannot be specified in the FQL query string. If navigation properties are specified in
1152 the FQL query string, the server shall fail the operation with HTTP status code 400 "Bad Request". This is
1153 motivated by the fact that FQL is a query language that remains local with the set of instances and by the
1154 desire to allow servers that internally use generic operations to pass the (decoded) FQL query string on
1155 without further processing it.

1156 Omitting the \$filter query parameter shall result in no additional restrictive filtering of instances in the
1157 instance collection.

1158 A \$filter query parameter that is specified with no value shall result in including no instances from the
1159 instance collection.

1160 Examples:

- 1161 (not specified)
- 1162 no additional restrictive instance filtering takes place
- 1163 `$filter=`
- 1164 includes no instances
- 1165 `$filter=Type='LAN'%20AND%20ErrorRate%3E0`
- 1166 specifies the FQL query string "`Type='LAN' AND ErrorRate>0`" and causes only instances
- 1167 with properties `Type = "LAN"` and `ErrorRate > 0` to be included.
- 1168 The reserved characters "=" and single quote (') in the FQL query string are not percent-
- 1169 encoded because they do not occur in literals of the FQL query string and are therefore
- 1170 considered delimiters.
- 1171 The blank and ">" characters are not allowed in resource identifiers and are therefore percent-
- 1172 encoded.
- 1173 `$filter=Description='a%2Cb%3D0'`
- 1174 specifies the FQL query string "`Description='a,b=0'`" and causes only instances with
- 1175 property `Description = "a,b=0"` to be included.
- 1176 The first occurrence of the reserved character "=" in the FQL query string (right after
- 1177 `Description`) is not percent-encoded because it does not occur in literals of the FQL query string
- 1178 and is therefore considered a delimiter.
- 1179 The second occurrence of the reserved character "=" and the reserved character "," in the FQL
- 1180 query string (in the `Description` value) are percent-encoded because they occur in a literal of the
- 1181 FQL query string and are therefore considered data.

1182 **6.5.5 \$max (limit number of collection members in result)**

- 1183 The `$max` query parameter limits the number of members in any retrieved collections to the specified
- 1184 number.
- 1185 If there are members in excess of that maximum number, the server shall return the collection in paged
- 1186 mode. Note that a server may choose to return the collection in paged mode also when the specified
- 1187 maximum number of members is not exceeded. For details on paging of collections, see 7.3.8.
- 1188 The format of this query parameter is defined by the following ABNF:
- 1189

```
query-parameter =/ max-query-parm
```
- 1190

```
max-query-parm = "$max=" max-members
```
- 1191

```
max-members = nonNegativeDecimalInteger
```
- 1192
- 1193
- 1194 Where:
- 1195 • The reserved characters "\$" and "=" in the literals of these ABNF rules shall be considered
 - 1196 delimiters for purposes of percent-encoding (see 6.3)
 - 1197 • `max-members` specifies the maximum number of collection members.
- 1198 The `$max` query parameter shall not be repeated in a resource identifier.

- 1199 Omitting the \$max query parameter indicates that there is no maximum number specified.
- 1200 Specifying the \$max query parameter with a value of 0 indicates that a collection with no members shall
1201 be returned.
- 1202 Note that a server may choose to use paging also when the no maximum is specified.
- 1203 Examples:
- 1204 (not specified)
- 1205 no maximum is specified for the number of members in the collection result.
- 1206 \$max=0
- 1207 number of members in the collection result is limited to no more than 0 (that is, the collection is
1208 empty).
- 1209 \$max=10
- 1210 number of members in the collection result is limited to no more than 10.

1211 6.5.6 \$methods (subset method links in result)

1212 The \$methods query parameter subsets the method invocation links any instances or instance collections
1213 to only those for the specified set of method names.

1214 The format of this query parameter is defined by the following ABNF:

```
1215 query-parameter =/ methods-query-parm
1216
1217 methods-query-parm = "$methods=" [ method-list ]
1218
1219 method-list = method-spec *( "," method-spec )
1220
1221 method-spec = [ nav-path "." ] method-name
```

1222 Where:

- 1223 • The reserved characters "\$", "=", and "," in the literals of these ABNF rules shall be considered
1224 delimiters for purposes of percent-encoding (see 6.3). Note that the character "." used in the in
1225 the literals of these ABNF rules is an unreserved character.
- 1226 • method-name is the name of a method (without parenthesis or any method parameters)
- 1227 • nav-path is the navigation path to the instances whose method invocation links are to be
1228 subsetted. nav-path and the concept of a navigation path is described in 5.6. Any reserved
1229 characters in the navigation path (that is, "[" and "]") shall be considered delimiters for purposes
1230 of percent-encoding (see 6.3). Note that the character "." in the navigation path is an
1231 unreserved character.

1232 The \$methods query parameter may be repeated in a resource identifier, see 6.5. If repeated, the
1233 effective method list shall be the combined method list of all occurrences of the \$methods query
1234 parameter.

1235 Omitting the \$methods query parameter shall result in not excluding any method invocation links.

1236 A \$methods query parameter that is specified with no value shall result in including no method invocation
1237 links in the instances, instance collections or instances in the instance collections.

1238 This query parameter may be specified only in resource identifiers of instance resources or instance
 1239 collection resources. If specified in resource identifiers of instance collection resources, it applies to the
 1240 instance collection itself and to all instances in the collection.

1241 Any navigation path used to identify method invocation links shall also be specified in the \$expand query
 1242 parameter. This ensures that the instances of such links are part of the retrieved instance
 1243 representations. If this condition is not met, the consumer shall fail the operation with HTTP status code
 1244 400 "Bad Request".

1245 Duplicate and invalid method names shall be ignored. Invalid method names are names of methods that
 1246 are not exposed by the creation class of an instance.

1247 Examples:

1248 (not specified)

1249 no method invocation links are excluded

1250 \$methods=

1251 no method invocation links are included

1252 \$methods=Start,Stop

1253 only the method invocation links for methods "Start" and "Stop" are included

1254 **6.5.7 \$pagingtimeout (specify inactivity timeout for paged retrieval)**

1255 The \$pagingtimeout query parameter specifies a duration after which a server may close a sequence of
 1256 paged retrievals of subset collections if there is no retrieval activity on that sequence. This duration is
 1257 referred to as *paging timeout*. For details, see 7.3.8.

1258 The format of this query parameter is defined by the following ABNF:

```
1259 query-parameter =/ pagingtimeout-query-param
1260
1261 pagingtimeout-query-param = "$pagingtimeout=" duration
1262
1263 duration = nonNegativeDecimalInteger
```

1264 Where:

- 1265 • The reserved characters "\$" and "=" in the literals of these ABNF rules shall be considered
 1266 delimiters for purposes of percent-encoding (see 6.3)
- 1267 • *duration* is the duration of the paging timeout in seconds. A value of 0 specifies that there is
 1268 no paging timeout (that is, an infinite paging timeout)

1269 The \$pagingtimeout query parameter shall not be repeated in a resource identifier.

1270 Omitting the \$pagingtimeout query parameter shall result in using the default paging timeout of the server
 1271 (see 7.12).

1272 The allowable values for the paging timeout clients may specify with the \$pagingtimeout query parameter
 1273 can be discovered by clients through the "minimumpagingtimeout" and "maximumpagingtimeout"
 1274 attributes of the server entry point resource (see 7.12).

1275 Examples:

1276 (not specified)

1277 default paging timeout of the server is used

1278 `$pagingtimeout=0`

1279 no paging timeout is used (infinite paging timeout)

1280 `$pagingtimeout=30`

1281 a paging timeout of 30 seconds is used

1282 **6.5.8 \$properties (subset properties in result)**

1283 The `$properties` query parameter subsets the properties in any retrieved instance representations to only
1284 the specified set of properties. This is semantically equivalent to acting on a different resource that is a
1285 subset of the full resource.

1286 The format of this query parameter is defined by the following ABNF:

```
1287 query-parameter =/ properties-query-param
1288
1289 properties-query-param = "$properties=" [ property-list ]
1290
1291 property-list = property-spec *( "," property-spec )
1292
1293 property-spec = [ nav-path "." ] property-name
```

1294 Where:

- 1295 • The reserved characters "\$", "=", and "," in the literals of these ABNF rules shall be considered
1296 delimiters for purposes of percent-encoding (see 6.3). Note that the character "." used in the in
1297 the literals of these ABNF rules is an unreserved character.
- 1298 • `property-name` is the name of a property in the instances
- 1299 • `nav-path` is the navigation path to the instances whose properties are to be subsetted. `nav-`
1300 `path` and the concept of a navigation path is described in 5.6. Any reserved characters in the
1301 navigation path (that is, "[" and "]") shall be considered delimiters for purposes of percent-
1302 encoding (see 6.3). Note that the character "." in the navigation path is an unreserved character.

1303 The `$properties` query parameter may be repeated in a resource identifier, see 6.5. If repeated, the
1304 effective property list shall be the combined property list of all occurrences of the `$properties` query
1305 parameter.

1306 Omitting the `$properties` query parameter shall result in not excluding any properties.

1307 A `$properties` query parameter that is specified with no value shall result in including no properties in the
1308 retrieved instance representations.

1309 The order of property names specified in the query parameter is not relevant for the order of properties in
1310 the retrieved instance representations.

1311 This query parameter may be specified only in resource identifiers of instance resources or instance
1312 collection resources. If specified in resource identifiers of instance collection resources, it applies to all
1313 instances in the collection.

1314 Any navigation path used to identify properties shall also be specified in the `$expand` query parameter.
1315 This ensures that the instances of such properties are part of the retrieved instance representations. If
1316 this condition is not met, the consumer shall fail the operation with HTTP status code 400 "Bad Request".

1317 Duplicate and invalid property names shall be ignored. Invalid property names are names of properties
1318 that are not exposed by the creation class of an instance.

1319 Examples:

1320 (not specified)

1321 no properties are excluded

1322 \$properties=

1323 no properties are included

1324 \$properties=Name,Type

1325 only the properties "Name" and "Type" are included

1326 **6.5.9 \$refer (include references to target instances)**

1327 The \$refer query parameter may be used on operations that retrieve instances or instance collections and
1328 specifies a list of navigation paths. For details on navigation paths and the resulting navigation properties,
1329 see 5.6.

1330 The value of navigation properties included as a result of using the \$refer query parameter shall be a
1331 reference collection whose members are references to the target instances identified by the navigation
1332 path. That reference collection shall be represented as a ReferenceCollection payload element (see
1333 7.7.1) and shall be subject to paged retrieval (see 7.3.8).

1334 Navigation paths that refer to existing references (qualified or typed, scalar or array) can be used to
1335 subset these references in the retrieved instance representations by specifying *filter-class-name* in
1336 the navigation path (see 5.6).

1337 The format of the \$refer query parameter is defined by the following ABNF:

```
1338 query-parameter =/ refer-query-param
1339
1340 refer-query-param = "$refer=" [ refer-list ]
1341
1342 refer-list = nav-path *( "," nav-path )
```

1343 Where:

- 1344 • The reserved characters "\$", "=", and "," in the literals of these ABNF rules shall be considered
1345 delimiters for purposes of percent-encoding (see 6.3).
- 1346 • *nav-path* is a navigation path identifying target instances, as defined in 5.6. Any reserved
1347 characters in the navigation path (that is, "[" and "]") shall be considered delimiters for purposes
1348 of percent-encoding (see 6.3). Note that the character "." in the navigation path is an
1349 unreserved character.

1350 The \$refer query parameter may be repeated in a resource identifier, see 6.5. If repeated, the effective
1351 refer list shall be the combined refer list of all occurrences of the \$refer query parameter.

1352 Duplicate or invalid navigation path strings in the set of all navigation paths specified for the \$expand or
1353 \$refer query parameters shall cause the operation to fail with HTTP status code 400 "Bad Request".

1354 Examples:

1355 (not specified)
 1356 \$refer=
 1357 No navigation paths have been specified; no navigation properties will be included
 1358 \$refer=ACME_SystemDevice.PartComponent,ACME_HostedService.Service
 1359 include navigation properties named "ACME_SystemDevice.PartComponent" and
 1360 "ACME_HostedService.Service" in each retrieved instance (assuming both are valid for the
 1361 retrieved instance)
 1362 For more examples, see D.1.

1363

1364 **6.6 Resource identifiers of entry point resources**

1365 The server and listener entry point resources are the only resources in the CIM-RS protocol that have
 1366 well-known resource identifiers.

1367 The resource identifier of the server entry point resource of a server shall have the path component
 1368 defined by the following ABNF rule:

1369 `server-entry-point-path = "/cimrs" ["/"]`

1370 The resource identifier of the listener entry point resource of a listener shall have the path component
 1371 defined by the following ABNF rule:

1372 `listener-entry-point-path = "/cimrs" ["/"]`

1373 Examples:

- 1374 /cimrs
- 1375 http://acme.com/cimrs/

1376 **7 Resources, operations and payload elements**

1377 This clause defines the types of resources used in the CIM-RS protocol, the operations on these
 1378 resources, and the payload elements used in the protocol payload when performing these operations.

1379 **7.1 Overview**

1380 Table 2 shows an overview of all types of resources used in the CIM-RS protocol. A resource in the CIM-
 1381 RS protocol is anything that can be the target of an HTTP method.

1382 **Table 2 – Resource types in CIM-RS**

Resource Type	Description
Instance resource	A resource within a server that represents a modeled object in the managed environment
Instance creation resource	A resource within a server that represents the ability to create instance resources (and thus, managed objects)
Instance collection resource	A resource within a server or listener that represents a collection of instance resources
Instance enumeration resource	A resource within a server that represents the ability to enumerate instance resources by class

Reference collection resource	A resource within a server or listener that represents a collection of references (to instance resources)
Method invocation resource	A resource within a server that represents the ability to invoke methods defined in a class
Listener destination resource	A resource within a listener that can be used to deliver indications
Server entry point resource	The entry point resource of a server; representing capabilities of the server, and providing the starting point for discovering further resources
Listener entry point resource	The entry point resource of a listener, representing capabilities of the listener

1383 A combination of a particular HTTP method on a particular type of resource is termed an "operation" in
 1384 this document. For ease of reference by other documents, these operations have names. However, the
 1385 names of the operations do not show up in the protocol.

1386 Table 3 shows all operations used in the CIM-RS protocol, identified by their HTTP method and target
 1387 resource type.

Table 3 – CIM-RS operations

HTTP Method	Target Resource Type	Description
DELETE	Instance resource	see 0
GET	Instance resource	see 7.6.3
PUT	Instance resource	see 7.6.4
POST	Instance creation resource	see 7.5.1
GET	Reference collection resource	see 7.7.2
GET	Instance collection resource	see 7.8.2
GET	Instance enumeration resource	see 7.9.1
GET	Listener entry point resource	see 7.13.2
POST	Listener destination resource	see 7.11.2
GET	Server entry point resource	see 7.12.2
POST	Method invocation resource	see 7.10.3

1389 Most of the operations used in the CIM-RS protocol have protocol payload data either in the request
 1390 message, or in the response message, or both. These payload elements often correspond directly to
 1391 resources, but not always. This document defines these payload elements in a normative but abstract
 1392 way. CIM-RS payload representation specifications define how each of these payload elements is
 1393 represented, for details see clause 9. The payload elements have a name for ease of referencing
 1394 between documents, as shown in the first column of Table 4.

1395 Table 4 shows all payload elements used in the CIM-RS protocol.

Table 4 – CIM-RS payload elements

Payload Element	Meaning	Description
Instance	representation of an instance resource; that is, a modeled object in the managed environment	See 7.6.1
ReferenceCollection	representation of a reference collection resource containing an order-preserving list of references to instance resources	See 7.7.1

Payload Element	Meaning	Description
InstanceCollection	representation of an instance collection resource containing an order-preserving list of instance resources	See 7.8.1
MethodRequest	the data used to request the invocation of a method	See 7.10.1
MethodResponse	the data used in the response of the invocation of a method	See 7.10.2
IndicationDeliveryRequest	the data used to request the delivery of an indication to a listener	See 7.11.1
ServerEntryPoint	representation of the server entry point resource of a WBEM server, describing protocol-level capabilities of the server, and providing resource identifiers for performing certain operations	See 7.12.1
ListenerEntryPoint	representation of the listener entry point resource of a WBEM listener, describing protocol-level capabilities of the listener	See 7.13.1
ErrorResponse	the data used in an error response to any request	See 7.3.6

1397

1398 **7.2 Description conventions**

1399 **7.2.1 Datatypes used in payload element definitions**

1400 This subclause defines the datatypes used in the definition of the attributes of payload elements. In order
 1401 to distinguish these kinds of datatypes from CIM datatypes, they are termed "payload datatypes". Payload
 1402 datatypes are used as a description mechanism for this document and for any payload representation
 1403 specifications.

1404 The representation of values of payload datatypes is defined in payload representation specifications; for
 1405 details see clause 9.

1406 **Table 5 – Datatypes used in payload elements**

Payload datatype	Description		
String	a string of UCS characters, or Null		
Integer	an integer value, or Null		
MethodLink	a complex type for method invocation links, containing the following child attributes:		
	Attribute	Payload datatype	Description
	name	String	name of the method (without any parenthesis or method parameters)
	class	String	name of the implemented class exposing the method
	uri	URI	resource identifier of the method invocation resource (see 7.10)

Payload datatype	Description												
ElementValue	<p>a complex type for representing the value of a typed CIM element (such as properties, method parameters or method return values), and optionally its CIM datatype, containing the following child attributes:</p> <table border="1"> <thead> <tr> <th>Attribute</th> <th>Payload datatype</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>String</td> <td>name of the element</td> </tr> <tr> <td>value</td> <td>multiple</td> <td>value of the element, represented as defined by the payload representation specification. Reference properties and reference parameters need to be represented as defined for the URI payload datatype.</td> </tr> <tr> <td>type</td> <td>String</td> <td>identification of the CIM datatype of the element, using the type strings defined by the payload representation specification</td> </tr> </tbody> </table>	Attribute	Payload datatype	Description	name	String	name of the element	value	multiple	value of the element, represented as defined by the payload representation specification. Reference properties and reference parameters need to be represented as defined for the URI payload datatype.	type	String	identification of the CIM datatype of the element, using the type strings defined by the payload representation specification
Attribute	Payload datatype	Description											
name	String	name of the element											
value	multiple	value of the element, represented as defined by the payload representation specification. Reference properties and reference parameters need to be represented as defined for the URI payload datatype.											
type	String	identification of the CIM datatype of the element, using the type strings defined by the payload representation specification											
URI	a CIM-RS resource identifier, in the format defined in 6.1												
Instance	an Instance payload element, as defined in 7.6.1												

1407 The CIM datatype specified in the "type" child element of the ElementValue type allows infrastructure
 1408 components to represent element values in programming environments using strong types for the CIM
 1409 datatypes. This is expected to be used for WBEM client implementations as model-neutral client libraries.

1410 Representation of the "type" child element of the ElementValue payload datatype is optional for payload
 1411 representations. If a payload representation supports representation of the "type" child element, it shall be
 1412 present; otherwise, it shall be omitted. Note that this decision is made by the definition of a payload
 1413 representation, and not by an implementation of CIM-RS.

1414 7.2.2 Requirement levels used in payload element definitions

1415 This subclause defines the meaning of requirement levels used in the definition of the attributes of
 1416 payload elements.

1417 **Mandatory** The attribute shall be included in the payload element.

1418 **Conditional** The attribute shall be included in the payload element if the condition is
 1419 met. If the condition is not met, the attribute may be included in the
 1420 payload element at the discretion of the implementation.

1421 **ConditionalExclusive** The attribute shall be included in the payload element if the condition is
 1422 met. If the condition is not met, the attribute shall not be included in the
 1423 payload element.

1424 **Optional** The attribute may be included in the payload element at the discretion of
 1425 the implementation.

1426 7.2.3 Requirement levels used in operation definitions

1427 This subclause defines the meaning of requirement levels used in the descriptions of operations:

1428 **Mandatory** The operation shall be implemented. It is not expected that the
 1429 implementation of the operation is specific to a class or model.

1430 **Mandatory (class specific)** The implementation of the operation is specific to a class or model.
 1431 General infrastructure support for the operation (that is, functionality not
 1432 specific to a class or model) shall be implemented; the requirements for
 1433 implementing the operation for specific classes are defined elsewhere
 1434 (for example, in management profiles)

1435 7.2.4 CIM-RS operation description format

1436 The definition of operations in the following subclauses uses the following description fields:

1437 **Name:** The name of the operation.

1438 **Purpose:** A brief description of the purpose of the operation.

1439 **HTTP method:** The name of the HTTP method used to perform the operation (for
 1440 example, GET, PUT, POST, DELETE).

1441 **Target resource:** The resource that is identified as the target of the HTTP method, by
 1442 means of the Request-URI field (see [RFC2616](#)) and Host header field.

1443 **Query parameters:** The names of any query parameters that may be specified in the
 1444 resource identifier. Other query parameters shall not be specified by the
 1445 requester. If other query parameters are specified by the requester, they
 1446 shall be ignored by the responder, in order to provide for future
 1447 extensibility.

1448 **Request headers:** The names of any header fields that may be specified in the request
 1449 message. Other request headers shall not be specified by the requester.
 1450 If other query request headers are specified by the requester, they shall
 1451 be ignored by the responder, in order to provide for future extensibility.

1452 **Request payload:** The name of the payload element that shall be used in the entity body of
 1453 the request message. "None" means the entity body shall be empty.

1454 **Response headers:** The names of any header fields that may be specified in the response
 1455 message, separately for the success and failure case Other response
 1456 headers shall not be specified by the responder. If other query request
 1457 headers are specified by the responder, they shall be ignored by the
 1458 requester, in order to provide for future extensibility.

1459 **Response payload:** The name of the payload element that shall be used in the entity body of
 1460 the response message, separately for the success and failure case.
 1461 "None" means the entity body shall be empty.

1462 **Requirement:** The requirement level to implement the operation, as defined in 7.2.3.

1463 **Description:** A normative definition of the behavior of the operation, in addition to the
 1464 normative definitions stated in the previous description fields.

1465 **Example HTTP conversation:** An example HTTP request and HTTP response.

1466 7.3 Common behaviors for all operations

1467 7.3.1 Content negotiation

1468 Wbem clients, servers, and listeners shall support server-driven content negotiation as defined in
 1469 [RFC2616](#), based on the Accept request-header (defined in [RFC2616](#) and in 8.4.1), and the Content-Type
 1470 response header field (defined in [RFC2616](#) and in 8.4.2).

1471 Requirements for the media types used in these header fields are defined in 9.1.

1472 The entry point resources of server and listener can be retrieved in order to discover the supported set of
1473 CIM-RS payload representations, as described in 7.12.2 and 7.13.2.

1474 **7.3.2 Verifying the basis of resource modifications (EXPERIMENTAL)**

1475 **EXPERIMENTAL**

1476 The HTTP PUT method on an instance resource (see 7.6.4) takes an instance with the new property
1477 values as input. The CIM-RS protocol provides for a means to verify for a server whether the current state
1478 of the resource is still the same as when the client retrieved the resource as a basis for the modifications.

1479 This may be achieved by using the value of the CIM Generation property (defined in ACME_Element) as
1480 an entity tag with the ETag and If-Match HTTP header fields, as described in 8.4.3 and 8.4.4.

1481 This ability is part of the optional entity tagging feature (see 7.4.1).

1482 **EXPERIMENTAL**

1483 **7.3.3 Caching of responses**

1484 Caching of responses from servers and listeners is described in [RFC2616](#). This document does not
1485 define any additional constraints or restrictions on caching.

1486 Note that any use of the HTTP GET method in the CIM-RS protocol is safe and idempotent, and that any
1487 use of the HTTP PUT method in the CIM-RS protocol is idempotent.

1488 Implementing the entity tagging feature (see 7.4.1) improves cache control.

1489 **7.3.4 Success and failure**

1490 Operations performed within the CIM-RS protocol shall either succeed or fail. There is no concept of
1491 "partial success".

1492 If an operation succeeds, it shall return its output data to the operation requester and shall not include any
1493 errors .

1494 If an operation fails, it shall return an error to the operation requester (see 7.3.6) and no output data.

1495 For example, if an instance collection retrieval operation were able to return some, but not all, instances
1496 successfully, then the operation fails without returning any instances.

1497 When using paged retrieval, each retrieval operation within a paged retrieval stream is considered a
1498 separate operation w.r.t. success and failure.

1499 **7.3.5 Errors**

1500 Errors at the CIM-RS protocol level are returned as HTTP status codes. The definition of HTTP status
1501 codes defined in [RFC2616](#) is the basis for each operation, and the operation descriptions in this
1502 document specify any additional constraints on the use of HTTP status codes.

1503 Extended error information is returned as an ErrorResponse payload element (see 7.3.6) in the entity
1504 body. For details about its usage, see the operation descriptions in clause 7.

1505 7.3.6 ErrorResponse payload element

1506 An ErrorResponse payload element represents the data used in an error response to any request.

1507 An ErrorResponse payload element shall have the following attributes:

1508 **Table 6 – Attributes of an ErrorResponse payload element**

Attribute name	Payload datatype	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value "errorresponse"
self	URI	Mandatory	resource identifier of the resource targeted by the HTTP method that failed
httpmethod	String	Mandatory	name of the HTTP method that failed
statuscode	Integer	Optional	CIM status code
statusdescription	String	Optional	CIM status description
errors	Instance []	Mandatory	order-preserving list of representations of zero or more embedded instances of class CIM_Error defined in the CIM Schema published by DMTF, with attribute "self" omitted, each specifying an error message

1509

1510 7.3.7 Consistency model

1511 The operations of the CIM-RS protocol shall conform to the consistency model defined in [DSP0223](#).

1512 7.3.8 Paging of collections

1513 Client and servers shall support the *paging of collections* returned to clients as described in this
1514 subclause.

1515 An instance collection contains an order-preserving list of instance representations). When a
1516 representation of an instance collection is returned to a client, the server may choose to use paging for
1517 the instance collection, at the server's discretion.

1518 A reference collection contains an order-preserving list of references to instances. When a representation
1519 of a reference collection is returned to a client, the server may choose to use paging for the reference
1520 collection, at the server's discretion.

1521 If the server does not use paging for a collection, the "next" attribute of that collection shall be omitted.

1522 If the server uses paging for a collection, its "next" attribute shall reference a collection resource that
1523 contains the next subset of collection members. That next subset collection may again contain only a
1524 subset of the remaining members, and so forth. The last subset collection has no "next" attribute,
1525 indicating that it is the last one of the sequence of subset collections.

1526 The members in each subset collection form an order-preserving list, and appending the lists of these
1527 subset collections in the order of their "next" links shall reconstruct the original order of members in the
1528 entire collection. In other words, the order of members in a collection is maintained when paging is used
1529 to retrieve the collection.

1530 As a result, any InstanceCollection payload element (see 7.8.1) or ReferenceCollection payload element
1531 (see 7.7.1) is self-describing w.r.t. whether it contains the last (or possibly only) set of members, or other
1532 subsets are following; and the subdivision of the complete set of instances into subset collections always

1533 happens at a granularity of complete instances (that is, instances are never broken apart to be returned in
1534 separate subset collections).

1535 Instance collection and reference collection resources can be retrieved directly using the HTTP GET
1536 method. Instance collections and reference collections can also be part of instances (for example, when
1537 using the \$expand or \$refer query parameters, see 5.6). If an instance (being retrieved directly, or being
1538 part of an instance collection that is retrieved) contains instance collections or reference collections, these
1539 nested collections may also be paged, at arbitrary nesting depth. Servers may choose to page or not to
1540 page the collections in a result independently of each other.

1541 Clients and servers shall support paging of collections for the following operations:

1542 **Table 7 – Operations supporting paging of collections**

HTTP Method	Target Resource Type	Retrieved Resource Representation	Description
GET	Instance resource	instance	see 7.6.3
GET	Reference collection resource	reference collection	see 7.7.2
GET	Instance collection resource	instance collection	see 7.8.2
GET	Instance enumeration resource	instance collection	see 7.9.1

1543 Clients may use the \$max query parameter (see 6.5.5) to limit the number of members in each returned
1544 (subset) collection.

1545 Each returned (subset) collection shall contain any number of members between one and the maximum
1546 specified with the \$max query parameter (if specified). The number of members in a collection may
1547 change between any two subset collections (belonging to the same or different entire collection, or
1548 operation). As a result, the number of members in a collection is not a safe indicator for a client that there
1549 are remaining members; only the presence of the "next" attribute is a safe indicator for that.

1550 Because the server decides about whether or not to page any collections, from a client's perspective the
1551 resource identifier of a collection resource sometimes references the entire collection, and sometimes
1552 only the first subset collection. As a result, the resources referenced by such resource identifiers
1553 represent *possibly paged collections*.

1554 The resource identifiers of the set of subset collections representing a complete collection shall all be
1555 distinct. Servers shall represent the state of retrieval progress within a sequence of subset collections in
1556 the resource identifiers of the subset collections.

1557 Servers should implement ceasing of subset collection resources. If a server implements ceasing of
1558 subset collection resources, successfully retrieved subsequent subset collections (that is, second to last)
1559 shall cause the retrieved subset collection resource to cease existence, and subsequent requests to
1560 retrieve that subset collection resource shall be rejected with HTTP status code 404 "Not Found".

1561 The first subset collection of a sequence shall not cease existence as a result of being successfully
1562 retrieved, when the server implements ceasing of subset collection resources (however, it may cease
1563 existence for other reasons, such as ceasing of the represented managed object). Separate retrieval
1564 requests for the entire and first subset collection shall be treated independently by the server (regardless
1565 of whether these requests come from the same or different clients, and regardless of whether a request is
1566 a repetition of an earlier request). As a result, each successful retrieval request of the first subset
1567 collection opens a new sequence of paged retrievals for the remaining subset collections.

1568 Clients and servers may support the continue on error feature (see 7.4.2). Clients that support the
1569 continue on error feature may request continuation on error for paged retrievals by specifying the
1570 \$continueonerror query parameter (see 6.5.2). If a retrieval request results in an error, the client has
1571 request continuation on error, and the server supports the continue on error feature, the server shall not
1572 close the sequence of retrievals. Otherwise, the server shall close the sequence of retrievals, if a retrieval

1573 request results in an error. For details on this behavior, see the description of "continuation on error" of
1574 pulled enumerations in [DSP0223](#).

1575 Servers should close a sequence of paged retrievals after some time of inactivity on that sequence, even
1576 if the client has not retrieved the sequence exhaustively. Clients may use the \$pagingtimeout query
1577 parameter (see 6.5.7) to specify the minimum duration the server is obliged to keep a sequence of paged
1578 subset collections open after retrieval of a subset collection. If the \$pagingtimeout query parameter is not
1579 specified, the server default shall be used, which is indicated in the "defaultPagingTimeout" attribute of
1580 the server entry point resource (see 7.12). For details on this behavior, see the description of "operation
1581 timeout" of pulled enumerations in [DSP0223](#).

1582 The concept of paging collections as described in this subclause is consistent with pulled enumerations
1583 as defined in [DSP0223](#), so that it fits easily with servers that support the semantics of pulled
1584 enumerations in their implementation.

1585 Servers that support pulled enumerations in their implementation can achieve to be entirely stateless
1586 w.r.t. paging collections, by maintaining the entire state data of the paging progress in the enumeration
1587 context value, and by representing the enumeration context value in the resource identifiers of
1588 subsequent (second to last) subset collections. Binary data in an enumeration context value can for
1589 example be represented using a base64url encoding (see [RFC4648](#)), typically without any "=" padding
1590 characters at the end.

1591 For more details on pulled enumerations and the concept of enumeration context values, see [DSP0223](#).

1592 NOTE: The use of HTTP range requests as defined in [RFC2616](#) has been considered and dismissed, because the
1593 semantics of an ordered sequence of items that can be accessed by item number cannot be provided by
1594 implementations that support the opaque server-defined enumeration context values mandated by [DSP0223](#).

1595 **7.4 Optional features of the CIM-RS protocol**

1596 This subclause defines optional features for the implementation of the CIM-RS protocol.

1597 **7.4.1 Entity tagging feature**

1598 Implementation of the entity tagging feature in servers and clients provides for verifying the basis of
1599 resource modifications and thus for improved consistency control in instance modifications (see 7.3.7)
1600 and for improved cache control (see 7.3.3).

1601 Implementation of the entity tagging feature is optional for clients and servers, independently.

1602 Implementation of the entity tagging feature in a server is indicated through the "entitytagging" attribute in
1603 the server entry point resource (see 7.12).

1604 **7.4.2 Continue on error feature**

1605 Implementation of the continue on error feature in servers provides clients with the possibility to request
1606 continuation of a sequence of paged retrievals in case of error. For details on paged retrieval, see 7.3.8.

1607 Implementation of the continue on error feature is optional for clients and servers, independently.

1608 Implementation of the continue on error feature in a server is indicated through the "continueonerror"
1609 attribute in the server entry point resource (see 7.12).

1610 **7.5 Instance creation resource**

1611 An instance creation resource represents the ability to create instance resources.

1612 As defined in 7.14, a server exposes one instance creation resource for each namespace that is
1613 supported for access by the CIM-RS protocol; its resource identifier is available through the "creation"

1614 attribute of the corresponding entry of the "namespaces" array attribute of the server entry point resource
1615 (see 7.11).

1616 7.5.1 POST

1617 **Purpose:** Creates an instance resource

1618 **HTTP method:** POST

1619 **Target resource:** Instance creation resource (see 7.5)

1620 **Query parameters:** \$class

1621 **Request headers:** Host, Content-Length, Content-Type, X-CIMRS-Version

1622 **Request payload:** Instance (see 7.6.1), without the "self" and "methods" attributes

1623 **Response headers (success):** Date, Location, X-CIMRS-Version

1624 **Response payload (success):** None

1625 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

1626 **Response payload (failure):** ErrorResponse (see 7.3.6)

1627 **Requirement:** Mandatory (class specific)

1628 **Description:**

1629 The HTTP POST method on an instance creation resource creates an instance of the specified class
1630 in the namespace of the targeted instance creation resource. The initial property values for the new
1631 instance are defined in an instance representation in the payload. On return, the Location header
1632 specifies the resource identifier of the newly created instance.

1633 The target resource identifier for this operation is specific to a namespace and can be obtained
1634 through the "creation" attribute of the corresponding entry of the "namespaces" array attribute of the
1635 server entry point resource (see 7.12). The entry for the desired namespace can be selected upfront
1636 by inspecting its "name" attribute. The desired class is specified as query parameter \$class (see
1637 6.5.1); it is required to be specified. If it is not specified, the server shall fail the operation with HTTP
1638 status code 404 "Not Found".

1639 The new instance shall have a creation class that is the class specified in the `class` query
1640 parameter in the namespace of the targeted instance creation resource.

1641 The set of properties to be initialized in the new instance by the server is the set of all properties
1642 exposed by the creation class.

1643 Properties specified in the Instance payload element represent client-supplied initial values for the
1644 new instance.

1645 Properties specified in the Instance payload element that are not properties exposed by the creation
1646 class shall cause the server to fail the operation with HTTP status code 403 "Forbidden". Properties
1647 specified in the Instance payload element that are not client-initializable shall cause the server to fail
1648 the operation with HTTP status code 403 "Forbidden".

1649 Client-initializable properties shall be initialized as specified for the property in the Instance payload
1650 element (including initializing the property to Null), or if the property is not specified in the Instance
1651 payload element, to the class-defined default value of the property, or to Null if no such default value
1652 is defined.

1653 Any other properties of the instance shall be initialized as defined by the implementation, taking into
1654 account any requirements on the initial values defined in the model.

1655 If the resulting initial values would violate these requirements, the server shall fail the operation with
1656 HTTP status code 403 "Forbidden".

1657 The "self" link in the Instance payload element in the request message shall not be specified. If
1658 specified, the request shall be rejected with HTTP status code 400 "Bad Request".

1659 Any method invocation links in the Instance payload element in the request message shall not be
1660 specified. If specified, the request shall be rejected with HTTP status code 400 "Bad Request".

1661 On success, the entity body shall contain no payload element and the following HTTP status code
1662 shall be returned:

- 1663 • 201 "Created": The "Location" header field is set to the resource identifier of the newly
1664 created instance

1665 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
1666 the following HTTP status codes shall be returned:

- 1667 • 400 "Bad Request": Requirements on the request payload element were not satisfied (for
1668 example, "self" link or method invocation links were specified)
- 1669 • 403 "Forbidden": Properties specified in the Instance payload element are not client-
1670 initializable, are not properties exposed by the creation class of the new instance, or the
1671 resulting initial values would violate requirements defined in the model
- 1672 • 404 "Not Found": Target instance creation resource does not exist, for example because
1673 the \$class query parameter is not specified, or because it specifies a non-existing class
- 1674 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
1675 method (see [RFC2616](#))

1676 **Example HTTP conversation (using JSON):**

1677 Request:

```
1678 POST /cimrs/root%2Fcimv2/create?class=ACME_RegisteredProfile HTTP/1.1
1679 Host: server.acme.com:5988
1680 Content-Length: XXX
1681 Content-Type: application/json;version=1.0.0
1682 X-CIMRS-Version: 1.0.0
1683
1684 {
1685   "kind": "instance",
1686   "class": "ACME_RegisteredProfile",
1687   "properties": {
1688     "RegisteredName": "Fan",
1689     "RegisteredOrganization": 2,
1690     "RegisteredVersion": "1.1.0"
1691   }
1692 }
```

1693 Response:

```
1694 HTTP/1.1 201 Created
1695 Date: Fri, 11 Nov 2011 10:11:00 GMT
```

```

1696 Location: http://server.acme.com:5988/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMT
1697 F%3AFan%3A1.1.0
1698 X-CIMRS-Version: 1.0.1
    
```

1699 NOTE: The key property InstanceID is not provided in the request, since key property values are determined
 1700 by the server. Other properties of the class (for example, Caption or Description) are initialized to their class-
 1701 defined default values, or to Null.

1702 **7.6 Instance resource**

1703 An instance resource represents a managed object in the managed environment.

1704 Because CIM-RS is model-neutral, it defines how instances are exposed as instance resources. A model
 1705 defines how managed objects are modeled as instances, by defining classes. In combination, this defines
 1706 how managed objects are represented as REST instance resources. For details, see 5.5.

1707 **7.6.1 Instance payload element**

1708 An Instance payload element is the representation of an instance resource (and thus, of a managed
 1709 object in the managed environment) in the protocol.

1710 Unless otherwise constrained, an Instance payload element shall have the attributes defined in Table 8.

1711 **Table 8 – Attributes of an Instance payload element**

Attribute name	Payload datatype	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value "instance"
self	URI	Mandatory	resource identifier of the represented instance
class	String	Mandatory	name of the creation class of represented instance
properties	ElementValue []	Conditional	unordered set of properties (see 7.2.1), representing all or a subset of the properties of the instance resource, including derived properties added via the \$refer query parameter (see 6.5.9) Condition: The payload element includes properties
methods	MethodLink []	Conditional	unordered set of method invocation links (see 7.2.1), representing a subset or the entire set of method invocation links for instance methods of the represented instance. Condition: The payload element includes method invocation links

1712 The following requirements apply to the child attributes of the "properties" attribute, if present:

- 1713 • the "name" and "value" child attributes shall be present
- 1714 • the "type" child attribute shall be present if the payload representation supports the
 1715 representation of the CIM datatype in element values, and shall be omitted otherwise

1716 The following requirements apply to the child attributes of the "methods" attribute, if present:

- 1717 • the "name" and "uri" child attributes shall be present

1718

1719 **7.6.2 DELETE**

- 1720 **Purpose:** Deletes an instance resource
- 1721 **HTTP method:** DELETE
- 1722 **Target resource:** Instance resource (see 7.6)
- 1723 **Query parameters:** None
- 1724 **Request headers:** Host, X-CIMRS-Version
- 1725 **Request payload:** None
- 1726 **Response headers (success):** Date, X-CIMRS-Version
- 1727 **Response payload (success):** None
- 1728 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version
- 1729 **Response payload (failure):** ErrorResponse (see 7.3.6)
- 1730 **Requirement:** Mandatory (class specific)

1731 **Description:**

1732 The HTTP DELETE method on an instance resource deletes the instance resource.

1733 On success, the entity body shall contain no payload element and the following HTTP status code
1734 shall be returned:

- 1735
- 204 "No Content"

1736 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
1737 the following HTTP status codes shall be returned:

- 1738
- 404 "Not Found": Target instance resource does not exist
 - any other 4xx (client error) or 5xx (server error) HTTP status code permissible for this
1739 HTTP method (see [RFC2616](#))
1740

1741 **Example HTTP conversation (using JSON):**

1742 Request:

```
1743 DELETE /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0 HTTP/1.1
1744 Host: server.acme.com:5988
1745 X-CIMRS-Version: 1.0.0
```

1746 Response:

```
1747 HTTP/1.1 204 No Content
1748 Date: Fri, 11 Nov 2011 10:11:00 GMT
1749 X-CIMRS-Version: 1.0.1
```

1750 **7.6.3 GET**

- 1751 **Purpose:** Retrieves an instance resource
- 1752 **HTTP method:** GET
- 1753 **Target resource:** Instance resource (see 7.6)

1754	Query parameters:	\$expand, \$refer, \$properties, \$methods, \$max, \$continueonerror,
1755		\$pagingtimeout
1756	Request headers:	Host, Accept, X-CIMRS-Version
1757	Request payload:	None
1758	Response headers (success):	Date, Content-Length, Content-Type, ETag, X-CIMRS-Version
1759	Response payload (success):	Instance (see 7.6.1)
1760	Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
1761	Response payload (failure):	ErrorResponse (see 7.3.6)
1762	Requirement:	Mandatory (class specific)

1763 **Description:**

1764 The HTTP GET method on an instance resource retrieves a representation of the specified instance
1765 resource.

1766 For details on the effects of the query parameters on the returned Instance payload element, see the
1767 descriptions of these query parameters in 6.5.

1768 Note that the returned Instance payload element may have navigation properties or expanded
1769 references as a result of using the \$expand or \$refer query parameters, as described in 5.6. Any
1770 collections in these navigation properties or expanded references may be paged (see 7.3.8), and the
1771 query parameters related to paged retrieval apply to those collections.

1772 On success, the entity body shall contain an Instance payload element (see 7.6.1) and one of the
1773 following HTTP status codes shall be returned:

- 1774 • 200 "OK": The entity body contains the response payload element
- 1775 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
1776 empty. This status code can only occur if the server supports conditional requests and the
1777 client has requested a conditional request

1778 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
1779 the following HTTP status codes shall be returned:

- 1780 • 404 "Not Found": Target instance resource does not exist
- 1781 • any other 4xx (client error) or 5xx (server error) HTTP status code permissible for this
1782 HTTP method (see [RFC2616](#))

1783 **Example HTTP conversation (using JSON):**

1784 **Request:**

```
1785 GET /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0 HTTP/1.1
1786 Host: server.acme.com:5988
1787 Accept: application/json;version=1.0
1788 X-CIMRS-Version: 1.0.0
```

1789 **Response:**

```
1790 HTTP/1.1 200 OK
1791 Date: Fri, 11 Nov 2011 10:11:00 GMT
```

```

1792 Content-Length: XXX
1793 Content-Type: application/json;version=1.0.1
1794 X-CIMRS-Version: 1.0.1
1795
1796 {
1797   "kind": "instance",
1798   "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0",
1799   "class": "ACME_RegisteredProfile",
1800   "properties": {
1801     "InstanceID": "DMTF:Fan:1.1.0",
1802     "RegisteredName": "Fan",
1803     "RegisteredOrganization": 2,
1804     "RegisteredVersion": "1.1.0",
1805     . . .
1806   },
1807   "methods": {
1808     "GetCentralInstances": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3
1809 A1.1.0/GetCentralInstances"
1810   }
1811 }

```

1812 7.6.4 PUT

1813	Purpose:	Modifies an instance resource (partially or fully)
1814	HTTP method:	PUT
1815	Target resource:	Instance resource (see 7.6)
1816	Query parameters:	\$properties
1817	Request headers:	Host, Content-Length, Content-Type, If-Match (EXPERIMENTAL), X-
1818		CIMRS-Version
1819	Request payload:	Instance (see 7.6.1)
1820	Response headers (success):	Date, X-CIMRS-Version
1821	Response payload (success):	None
1822	Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
1823	Response payload (failure):	ErrorResponse (see 7.3.6)
1824	Requirement:	Mandatory (class specific)
1825	Description:	

1826 The HTTP PUT method on an instance resource sets some or all property values of the specified
 1827 instance resource.

1828 Partial modification of an instance is achieved by specifying the desired subset of properties in the
 1829 resource identifier using the \$properties query parameter (see 6.5.8). Since query parameters are
 1830 part of the address of a resource (see [RFC2616](#)), this approach performs a full replacement of the
 1831 resource representing the partial instance, satisfying the idempotency requirement for the PUT
 1832 method demanded by [RFC2616](#).

1833 If the \$properties query parameter is not specified, the set of properties to be set is the set of all
1834 mutable properties of the target instance. If the \$properties query parameter is specified, the set of
1835 properties to be set is the set of properties specified in the \$properties query parameter. Properties
1836 specified in the \$properties query parameter that are not properties of the target instance shall cause
1837 the server to fail the operation with HTTP status code 404 "Not Found". Properties specified in the
1838 \$properties query parameter that are not mutable shall cause the server to fail the operation with
1839 HTTP status code 403 "Forbidden".

1840 Properties specified in the Instance payload element that are not to be set as previously defined,
1841 shall be tolerated and ignored, even when they are not properties of the target instance.

1842 Mutable properties that are to be set as previously defined shall be set as specified for the property
1843 in the Instance payload element (including setting the property to Null), or if the property is not
1844 specified in the Instance payload element, to the class-defined default value of the property, or to
1845 Null if no such default value is defined.

1846 NOTE: This behavior for properties that are to be set but not specified in the Instance payload element is
1847 consistent with CIM-XML ([DSP0200](#)). In contrast, generic operations ([DSP0223](#)) requires that the property is set
1848 to Null in this case, even when a non-Null default value for the property is defined in the class.

1849 Requirements on mutability of properties can be defined in the model. Key properties are always
1850 unmutable.

1851 The "self" link in the Instance payload element in the request message is optional. If specified, it shall
1852 reference the same resource as the target resource identifier.

1853 Any method invocation links in the Instance payload element in the request message should not be
1854 specified. If specified, they shall be ignored by the server.

1855 EXPERIMENTAL

1856 In addition, a server shall cause the PUT method to fail with HTTP status code 409 "Conflict" if an If-
1857 Match header field is provided, and the entity tag provided as its value does not match the current
1858 entity tag of the resource. See 7.4.1 for more details on verifying the basis for resource
1859 modifications.

1860 EXPERIMENTAL

1861 On success, the entity body shall contain no payload element and the following HTTP status code
1862 shall be returned:

- 1863 • 204 "No Content"

1864 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
1865 the following HTTP status codes shall be returned:

- 1866 • 403 "Forbidden": A property specified in the \$properties query parameter was unmutable
- 1867 • 404 "Not Found": Target instance resource does not exist; or the \$properties query
1868 parameter specifies properties that are not properties of the target instance
- 1869 • 409 "Conflict": Verification of the basis for resource modifications was requested by
1870 specifying an If-Match header field, and the entity tag specified in the If-Match header field
1871 did not match the current entity tag of the resource
- 1872 • any other 4xx (client error) or 5xx (server error) HTTP status code permissible for this
1873 HTTP method (see [RFC2616](#))

1874 Example HTTP conversation (using JSON) for the full replacement of an instance:**1875 Request:**

```
1876 PUT /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0 HTTP/1.1
1877 Host: server.acme.com:5988
1878 Content-Length: XXX
1879 Content-Type: application/json;version=1.0.0
1880 X-CIMRS-Version: 1.0.0
1881
1882 {
1883     "kind": "instance",
1884     "class": "ACME_RegisteredProfile",
1885     "properties": {
1886         "RegisteredName": "Fan",
1887         "RegisteredOrganization": 2,
1888         "RegisteredVersion": "1.1.1",
1889         "Caption": "A changed caption"
1890     }
1891 }
```

1892 Response:

```
1893 HTTP/1.1 200 OK
1894 Date: Fri, 11 Nov 2011 10:11:00 GMT
1895 X-CIMRS-Version: 1.0.1
```

1896 NOTE: In this example, it is assumed that all provided properties are mutable. The mutable properties not provided
1897 (for example, Description) are set to their class-defined default values or to Null. The value of the InstanceID key
1898 property remains unchanged, since key properties are never mutable.

1899 Example HTTP conversation (using JSON) for the partial replacement of an instance:**1900 Request:**

```
1901 PUT /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0?properties=Regist
1902 eredVersion,Caption HTTP/1.1
1903 Host: server.acme.com:5988
1904 Content-Length: XXX
1905 Content-Type: application/json;version=1.0.0
1906 X-CIMRS-Version: 1.0.0
1907
1908 {
1909     "kind": "instance",
1910     "class": "ACME_RegisteredProfile",
1911     "properties": {
1912         "RegisteredVersion": "1.1.1",
1913         "Caption": "A changed caption"
1914     }
1915 }
```

1916 Response:

```
1917 HTTP/1.1 200 OK
1918 Date: Fri, 11 Nov 2011 10:11:00 GMT
```

1919 X-CIMRS-Version: 1.0.1

1920 NOTE: In this example, it is assumed that all provided properties are mutable. Only the RegisteredVersion and
1921 Caption properties are set to their new values.

1922 7.7 Reference collection resource

1923 A reference collection resource represents an order-preserving list of references to instance resources.

1924 7.7.1 ReferenceCollection payload element

1925 A ReferenceCollection payload element is the representation of a reference collection resource in the
1926 protocol.

1927 Unless otherwise constrained, a ReferenceCollection payload element shall have the attributes defined in
1928 Table 9.

1929 **Table 9 – Attributes of an ReferenceCollection payload element**

Attribute name	Payload datatype	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value "referencecollection"
self	URI	Mandatory	resource identifier of the represented reference collection. (that is, only the returned portion if paged retrieval mode is used for the result)
next	URI	Mandatory	resource identifier of the next subset reference collection, if any remaining references are available. Otherwise, this attribute shall be omitted.
class	String	Mandatory	name of the common superclass of the creation classes of the instances referenced in the reference collection of the entire result, if such a common superclass exists. Otherwise, the empty string
references	URI []	Mandatory	order-preserving list of resource identifiers representing the references that are the members of this collection

1930 7.7.2 GET

1931 **Purpose:** Retrieves a reference collection resource

1932 **HTTP method:** GET

1933 **Target resource:** Reference collection resource (see 7.7)

1934 **Query parameters:** \$max, \$continueonerror, \$pagingtimeout

1935 **Request headers:** Host, Accept, X-CIMRS-Version

1936 **Request payload:** None

1937 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

1938 **Response payload (success):** ReferenceCollection (see 7.7.1)

1939 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

1940 **Response payload (failure):** ErrorResponse (see 7.3.6)

1941 **Requirement:** Mandatory (class specific)

1942 **Description:**

1943 The HTTP GET method on a reference collection resource retrieves a representation of the specified
1944 reference collection resource.

1945 The target resource identifier for this operation is typically discovered from the "next" attribute of
1946 reference collections that are returned in paged mode (see 7.3.8).

1947 For details on the effects of the query parameters on the returned ReferenceCollection payload
1948 element, see the descriptions of these query parameters in 6.5.

1949 Any retrieval of a reference collection may be paged (see 7.3.8).

1950 On success, the entity body shall contain a ReferenceCollection payload element (see 7.8.1) and
1951 one of the following HTTP status codes shall be returned:

- 1952 • 200 "OK": The entity body contains the response payload element
- 1953 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
1954 empty. This status code can only occur if the server supports conditional requests and the
1955 client has requested a conditional request

1956 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
1957 the following HTTP status codes shall be returned:

- 1958 • 404 "Not Found": Target reference collection resource does not exist. This includes the
1959 case where paged retrieval is used and the sequence of paged retrievals has been closed
1960 by the server
- 1961 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
1962 method (see [RFC2616](#))

1963 **Example HTTP conversation (using JSON):**

1964 Request:

```
1965 GET /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/refer/ACME_ElementConformsToProfile.ManagedElement/part/2 HTTP/1.1
1966 Host: server.acme.com:5988
1967 Accept: application/json;version=1.0
1968 X-CIMRS-Version: 1.0.0
1969
```

1970 Response:

```
1971 HTTP/1.1 200 OK
1972 Date: Fri, 11 Nov 2011 10:11:00 GMT
1973 Content-Length: XXX
1974 Content-Type: application/json;version=1.0.1
1975 X-CIMRS-Version: 1.0.1
1976
1977 {
1978   "kind": "referencecollection",
1979   "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/refer/ACME_ElementConformsToProfile.ManagedElement/part/2",
1980   "class": "ACME_Fan",
1981   "references": [
1982     "/cimrs/root%2Fcimv2/ACME_Fan/fan11",
1983
```

```

1984     "/cimrs/root%2Fcimv2/ACME_Fan/fan12"
1985     ]
1986     }
    
```

1987 In this example, a client had previously retrieved an ACME_RegisteredProfile instance for the DMTF Fan
 1988 Profile V1.1.0 and had requested the inclusion of a navigation property named
 1989 "ACME_ElementConformsToProfile.ManagedElement" by specifying
 1990 \$refer=ACME_ElementConformsToProfile.ManagedElement.

1991 The value of that navigation property is a reference collection, as it turns out, of ACME_Fan instances.
 1992 The server decided to return that reference collection in paged mode, and the first subset of 10 fan
 1993 references was part of the response to the original retrieval request. The representation of the collection
 1994 in that response included a "next" attribute for retrieving the next subset of the reference collection.

1995 What we see in the example above is the retrieval of that next subset, which happens to contain the
 1996 references to fans number 11 and 12, and no "next" attribute because this subset completed the
 1997 collection.

1998 **7.8 Instance collection resource**

1999 An instance collection resource represents an order-preserving list of instance resources, which are the
 2000 result of some operation such as instance enumeration or association traversal. An instance collection
 2001 resource in a response can be represented in its entirety, or in pages (see 7.3.8). If represented in its
 2002 entirety, the instance collection is embedded in the result and does not have a resource URI. If
 2003 represented in pages, the first page is embedded in the result and does not have a resource URI, and
 2004 any remaining pages have a resource URI specific to that page.

2005 **7.8.1 InstanceCollection payload element**

2006 An InstanceCollection payload element is the representation of an instance collection resource in the
 2007 protocol, both when represented in its entirety or when represented in pages.

2008 Unless otherwise constrained, an InstanceCollection payload element shall have the attributes defined in
 2009 Table 10.

2010 **Table 10 – Attributes of an InstanceCollection payload element**

Attribute name	Payload datatype	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value "instancecollection"
self	URI	Conditional	resource identifier of the represented instance collection page (second page or further). Condition: The instance collection is represented in pages, and this payload element does not represent the first page
next	URI	Conditional	resource identifier of the next instance collection page. Condition: There are remaining instances available in the overall instance collection
class	String	Mandatory	name of the common superclass of the creation classes of the instances in the overall instance collection, if such a common superclass exists. Otherwise, the empty string
instances	Instance []	Mandatory	order-preserving list of Instance payload elements (see 7.6.1) representing the instances in this page of the overall instance collection

2011 **7.8.2 GET**2012 **Purpose:** Retrieves the next page of a paged instance collection resource2013 **HTTP method:** GET2014 **Target resource:** Page of an instance collection resource (see 7.8)2015 **Query parameters:** \$max2016 **Request headers:** Host, Accept, X-CIMRS-Version2017 **Request payload:** None2018 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version2019 **Response payload (success):** InstanceCollection (see 7.8.1)2020 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version2021 **Response payload (failure):** ErrorResponse (see 7.3.6)2022 **Requirement:** Mandatory (class specific)2023 **Description:**2024 The HTTP GET method on page of an instance collection resource retrieves a representation of the
2025 specified page of the overall instance collection.2026 The target resource identifier for this operation is discovered from the "next" attribute of the previous
2027 page of the instance collection (see 7.3.8).2028 For details on the effects of the query parameters on the returned InstanceCollection payload
2029 element, see the descriptions of these query parameters in 6.5.2030 Note that the instances in the returned InstanceCollection payload element may have navigation
2031 properties or expanded references as a result of using the \$expand or \$refer query parameters, as
2032 described in 5.6. Any collections in these navigation properties or expanded references may be
2033 paged (see 7.3.8), and the query parameters related to paged retrieval apply to those collections.

2034 Any retrieval of an instance collection may be paged (see 7.3.8).

2035 On success, the entity body shall contain an InstanceCollection payload element (see 7.8.1) and one
2036 of the following HTTP status codes shall be returned:

- 2037
- 200 "OK": The entity body contains the response payload element
 - 304 "Not Modified": The validators matched on a conditional request; the entity body is empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request
- 2038
-
- 2039
-
- 2040

2041 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
2042 the following HTTP status codes shall be returned:

- 2043
- 404 "Not Found": Target instance collection resource page does not exist. This includes the case where paged retrieval is used and the sequence of paged retrievals has been closed by the server
 - any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP method (see [RFC2616](#))
- 2044
-
- 2045
-
- 2046
-
- 2047

2048 **Example HTTP conversation (using JSON):**2049 **Request:**

```

2050 GET /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/ACME_ReferencedPr
2051 ofile/Antecedent HTTP/1.1
2052 Host: server.acme.com:5988
2053 Accept: application/json;version=1.0
2054 X-CIMRS-Version: 1.0.0

```

2055 **Response:**

```

2056 HTTP/1.1 200 OK
2057 Date: Fri, 11 Nov 2011 10:11:00 GMT
2058 Content-Length: XXX
2059 Content-Type: application/json;version=1.0.1
2060 X-CIMRS-Version: 1.0.1
2061
2062 {
2063   "kind": "instancecollection",
2064   "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/ACME_Refer
2065 encedProfile/Antecedent",
2066   "class": "ACME_RegisteredProfile",
2067   "instances": [
2068     {
2069       "kind": "instance",
2070       "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0",
2071       "class": "ACME_RegisteredProfile",
2072       "properties": {
2073         "InstanceID": "DMTF:Fan:1.1.0",
2074         "RegisteredName": "Fan",
2075         "RegisteredOrganization": 2,
2076         "RegisteredVersion": "1.1.0",
2077         . . . ,
2078         "ACME_ReferencedProfile": {
2079           "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/AC
2080 ME_ReferencedProfile",
2081           "Dependent": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0
2082 .0/ACME_ReferencedProfile/Dependent"
2083         }
2084       },
2085       "methods": {
2086         "GetCentralInstances": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AF
2087 an%3A1.1.0/GetCentralInstances"
2088       }
2089     },
2090     . . .
2091   ]
2092 }

```

2093 In this example, the operation traverses from a starting instance of class ACME_RegisteredProfile to the
 2094 set of instances associated through the ACME_ReferencedProfile association, specifically its Antecedent
 2095 end.

2096 The returned set of instances is again of class ACME_RegisteredProfile and has a navigation property
 2097 named ACME_ReferencedProfile for navigating back.

2098 **7.9 Instance enumeration resource**

2099 An instance enumeration resource represents the ability to enumerate instances of a class (including
 2100 subclasses) in a namespace of a server, returning them as an instance collection.

2101 As defined in 7.14, a server exposes one instance enumeration resource; its resource identifier is
 2102 available through the "enumeration" attribute of the corresponding entry of the "namespaces" array
 2103 attribute of the server entry point resource (see 7.11).

2104 **7.9.1 GET**

2105 **Purpose:** Enumerates instance resources by class

2106 **HTTP method:** GET

2107 **Target resource:** Instance enumeration resource (see 7.9)

2108 **Query parameters:** \$class, \$filter, \$expand, \$refer, \$properties, \$methods, \$max,
 2109 \$continueonerror, \$pagingtimeout

2110 **Request headers:** Host, Accept, X-CIMRS-Version

2111 **Request payload:** None

2112 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

2113 **Response payload (success):** InstanceCollection (see 7.8.1)

2114 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2115 **Response payload (failure):** ErrorResponse (see 7.3.6)

2116 **Requirement:** Mandatory (class specific)

2117 **Description:**

2118 The HTTP GET method on an instance enumeration resource enumerates all instances of the
 2119 specified class (including instances of subclasses) in the namespace of the targeted instance
 2120 enumeration resource and returns an instance collection with representations of these instances.

2121 The target resource identifier for this operation is specific to a namespace and can be obtained
 2122 through the "enumeration" attribute of the corresponding entry in the "namespaces" array attribute of
 2123 the server entry point resource (see 7.11). The entry for the desired namespace can be selected
 2124 upfront by inspecting its "name" attribute. The desired class is specified as query parameter \$class
 2125 (see 6.5.1); it is required to be specified. If it is not specified, the server shall fail the operation with
 2126 HTTP status code 404 "Not Found".

2127 For details on the effects of the query parameters on the returned InstanceCollection payload
 2128 element, see the descriptions of these query parameters in 6.5.

2129 Note that the instances in the returned InstanceCollection payload element may have navigation
 2130 properties or expanded references as a result of using the \$expand or \$refer query parameters, as

2131 described in 5.6. Any collections in these navigation properties or expanded references may be
2132 paged (see 7.3.8), and the query parameters related to paged retrieval apply to those collections.

2133 Any retrieval of an instance collection may be paged (see 7.3.8)

2134 On success, the entity body shall contain an InstanceCollection payload element (see 7.8.1) and one
2135 of the following HTTP status codes shall be returned:

- 2136 • 200 "OK": The entity body contains the response payload element. This includes the case
2137 where the specified class and namespace exist, but the result set of instances is empty
- 2138 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
2139 empty. This status code can only occur if the server supports conditional requests and the
2140 client has requested a conditional request

2141 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
2142 the following HTTP status codes shall be returned:

- 2143 • 404 "Not Found": Target instance enumeration resource does not exist, for example
2144 because the `class` query parameter is not specified, or because it specifies a non-existing
2145 class. This includes the case where paged retrieval is used and the sequence of paged
2146 retrievals has been closed by the server
- 2147 • any other 4xx (client error) or 5xx (server error) HTTP status code permissible for this
2148 HTTP method (see [RFC2616](#))

2149 Example HTTP conversation:

2150 Request:

```
2151 GET /cimrs/root%2Fcimv2/enum?class=ACME_System HTTP/1.1
2152 Host: server.acme.com:5988
2153 Accept: application/json;version=1.0
2154 X-CIMRS-Version: 1.0.1
```

2155 Response:

```
2156 HTTP/1.1 200 OK
2157 Date: Fri, 11 Nov 2011 10:11:00 GMT
2158 Content-Length: XXX
2159 Content-Type: application/json;version=1.0.0
2160 X-CIMRS-Version: 1.0.0
2161
2162 {
2163   "kind": "instancecollection",
2164   "self": "/cimrs/root%2Fcimv2/enum?class=ACME_System",
2165   "class": "ACME_System",
2166   "instances": [
2167     {
2168       "kind": "instance",
2169       "self": "/cimrs/root%2Fcimv2/ACME_ComputerSystem/sys1",
2170       "class": "ACME_ComputerSystem",
2171       "properties": {
2172         "InstanceID": "sys1",
2173         "Name": "sys1",
2174         . . .
```

```

2175     },
2176     "methods": {
2177         "RequestStateChange": "/cimrs/root%2Fcimv2/ACME_ComputerSystem/sys1/Request
2178 StateChange"
2179     }
2180 },
2181 . . .
2182 ]
2183 }
    
```

2184 NOTE: This example assumes that ACME_ComputerSystem is a subclass of ACME_System.

2185 **7.10 Method invocation resource**

2186 A method invocation resource represents the ability to invoke a method defined in a class (static or non-
 2187 static). Non-static methods can be invoked on instances, using the method invocation resources available
 2188 through the "methods" attribute of an instance resource (see 7.6). Static methods can be invoked on
 2189 classes, using the method invocation resources available through the "staticmethods" attribute of the
 2190 corresponding entry of the "namespaces" array attribute of the server entry point resource (see 7.12).

2191 **7.10.1 MethodRequest payload element**

2192 A MethodRequest payload element is the representation of a request to invoke a method in the protocol.

2193 A MethodRequest payload element shall have the attributes defined in Table 11.

2194 **Table 11 – Attributes of a MethodRequest payload element**

Attribute name	Payload datatype	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value "methodrequest"
self	URI	Mandatory	resource identifier of the method resource
method	String	Mandatory	method name (without any parenthesis or method parameters)
parameters	ElementValue []	Conditional	unordered set of method input parameters. Condition: The payload element includes method input parameters

2195

2196 The following requirements apply to the child attributes of the "parameters" attribute, if present:

2197

- 2198 • the "name" and "value" child attributes shall be present
- 2199 • the "type" child attribute shall be present if the payload representation supports the representation of the CIM datatype in element values, and shall be omitted otherwise

2200 **7.10.2 MethodResponse payload element**

2201 A MethodResponse payload element is the representation of the response of a method invocation in the
 2202 protocol.

2203 A MethodResponse payload element shall have the attributes defined in Table 12.

2204

Table 12 – Attributes of a MethodResponse payload element

Attribute name	Payload datatype	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value "methodresponse"
self	URI	Mandatory	resource identifier of the method resource
method	String	Mandatory	method name (without any parenthesis or method parameters)
returnvalue	ElementValue	Mandatory	method return value
parameters	ElementValue []	Conditional	unordered set of method output parameters. Condition: The payload element includes method output parameters

2205 The following requirements apply to the child attributes of the "returnvalue" attribute:

- 2206 • the "name" child attribute shall be omitted
- 2207 • the "value" child attribute shall be present
- 2208 • the "type" child attribute shall be present if the payload representation supports the
- 2209 representation of the CIM datatype in element values, and shall be omitted otherwise

2210 The following requirements apply to the child attributes of the "parameters" attribute, if present:

- 2211 • the "name" and "value" child attributes shall be present
- 2212 • the "type" child attribute shall be present if the payload representation supports the
- 2213 representation of the CIM datatype in element values, and shall be omitted otherwise

2214

2215 7.10.3 POST

- 2216 **Purpose:** Invokes a method (static or non-static)
- 2217 **HTTP method:** POST
- 2218 **Target resource:** Method invocation resource (see 7.10)
- 2219 **Query parameters:** None
- 2220 **Request headers:** Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
- 2221 **Request payload:** MethodRequest (see 7.10.1)
- 2222 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version
- 2223 **Response payload (success):** MethodResponse (see 7.10.2)
- 2224 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version
- 2225 **Response payload (failure):** ErrorResponse (see 7.3.6)
- 2226 **Requirement:** Mandatory (class specific)
- 2227 **Description:**

2228 The HTTP POST method on a method invocation resource invokes a method defined in a class
2229 (extrinsic method).

2230 The method can be static or non-static:

- 2231 • Non-static methods can be invoked on instances, using the method invocation links available
2232 through the "methods" attribute of an instance resource (see 7.6). A method invocation link for a
2233 non-static method is specific to the instance the method is invoked on, and to the method.
- 2234 • Static methods can be invoked on classes, using the method invocation links available through
2235 the "staticmethods" attribute of the corresponding entry of the "namespaces" array attribute of
2236 the server entry point resource (see 7.12). A method invocation link for a static method is
2237 specific to the class the method is invoked on, the namespace of the class, and to the method.

2238 On success, the entity body shall contain a MethodResponse payload element (see 7.10.2) and one
2239 of the following HTTP status codes shall be returned:

- 2240 • 200 "OK": The entity body contains the response payload element

2241 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
2242 the following HTTP status codes shall be returned:

- 2243 • 404 "Not Found": Target method invocation resource does not exist
- 2244 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
2245 method (see [RFC2616](#))

2246 Note that the ErrorResponse payload element used on failure cannot represent method output
2247 parameters or a method return value.

2248 **Example HTTP conversation (using JSON) for invocation of non-static method:**

2249 Request:

```
2250 POST /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentralInstan
2251 ces HTTP/1.1
2252 Host: server.acme.com:5988
2253 Accept: application/json;version=1.0
2254 Content-Length: XXX
2255 Content-Type: application/json;version=1.0.0
2256 X-CIMRS-Version: 1.0.0
2257
2258 {
2259   "kind": " methodrequest",
2260   "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentral
2261 Instances",
2262   "method": "GetCentralInstances",
2263   "parameters": {
2264     "MaxNumber": 1000
2265   }
2266 }
```

2267 Response:

```
2268 HTTP/1.1 200 OK
2269 Date: Fri, 11 Nov 2011 10:11:00 GMT
2270 Content-Length: XXX
2271 Content-Type: application/json;version=1.0.1
```

```

2272 X-CIMRS-Version: 1.0.1
2273
2274 {
2275   "kind": " methodresponse",
2276   "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentral
2277 Instances",
2278   "method": "GetCentralInstances",
2279   "returnvalue": 0,
2280   "parameters": {
2281     "ActualNumber": 25
2282   }
2283 }
    
```

2284 **7.11 Listener destination resource**

2285 A listener destination resource in a listener represents the ability to deliver an indication to the listener.

2286 NOTE: Listener destination resources in listeners should not be confused with modeled objects in servers that may
 2287 are also called "listener destinations" in some models (for example, in the event model of the CIM Schema), but
 2288 merely describe the information in the server about the location of the listener.

2289 **7.11.1 IndicationDeliveryRequest payload element**

2290 An IndicationDeliveryRequest payload element is the representation of a request to deliver an indication
 2291 to a listener in the protocol.

2292 An IndicationDeliveryRequest payload element shall have the attributes defined in Table 13.

2293 **Table 13 – Attributes of an IndicationDeliveryRequest payload element**

Attribute name	Payload datatype	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value "indicationdeliveryrequest"
self	URI	Mandatory	resource identifier of the listener destination resource
indication	Instance	Mandatory	an instance of a class that is an indication, specifying the indication to be delivered, with attribute "self" omitted

2294

2295 **7.11.2 POST**

2296 **Purpose:** Delivers an indication to a listener

2297 **HTTP method:** POST

2298 **Target resource:** Listener destination resource (see 7.11)

2299 **Query parameters:** None

2300 **Request headers:** Host, Accept, Content-Length, Content-Type, X-CIMRS-Version

2301 **Request payload:** IndicationDeliveryRequest (see 7.11.1)

2302 **Response headers (success):** Date, X-CIMRS-Version

2303 **Response payload (success):** None

2304 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2305 **Response payload (failure):** ErrorResponse (see 7.3.6)

2306 **Requirement:** Mandatory

2307 **Description:**

2308 The HTTP POST method on a listener destination resource delivers an indication to the listener
2309 specified in that resource.

2310 For implementations supporting the event model defined in the CIM Schema published by DMTF, the
2311 target resource identifier for this operation is the value of the Destination property of
2312 CIM_ListenerDestination instances that indicate the CIM-RS protocol in their Protocol property. For
2313 details, see the *DMTF Indications Profile* ([DSP1054](#)).

2314 On success, the entity body shall contain no payload element and one of the following HTTP status
2315 codes shall be returned:

- 2316 • 200 "OK"

2317 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
2318 the following HTTP status codes shall be returned:

- 2319 • 404 "Not Found": Target listener destination resource does not exist
- 2320 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
2321 method (see [RFC2616](#))

2322 **Example HTTP conversation (using JSON):**

2323 Request:

```
2324 POST /cimrs/dest1 HTTP/1.1
2325 Host: listener.acme.com:5988
2326 Accept: application/json;version=1.0
2327 Content-Length: XXX
2328 Content-Type: application/json;version=1.0.0
2329 X-CIMRS-Version: 1.0.1
2330
2331 {
2332   "kind": "indicationdeliveryrequest",
2333   "self": "/cimrs/dest1",
2334   "indication": {
2335     "kind": "instance",
2336     "class": "ACME_AlertIndication",
2337     "properties": {
2338       "AlertType": 4,
2339       "PerceivedSeverity": 5,
2340       "ProbableCause": 42,
2341       "Message": "BOND0007: Some error happened, rc=23.",
2342       "MessageArguments": [ "23" ],
2343       "MessageID": "BOND0007",
2344       "OwningEntity": "ACME"
2345     }
2346   }
2347 }
```

```
2346     }
2347     }
```

2348 Response:

```
2349     HTTP/1.1 204 No Content
2350     Date: Fri, 11 Nov 2011 10:11:00 GMT
2351     X-CIMRS-Version: 1.0.0
```

2352 **7.12 Server entry point resource**

2353 A server entry point resource describes protocol-level capabilities of a server, and provides a starting
2354 point for discovering further resources in the server.

2355 The representation of the server entry point resource provides some server capabilities, the list of
2356 namespaces for which the server supports the CIM-RS protocol, and resource identifiers of resources that
2357 provide for performing operations:

- 2358 • instance enumeration resource: A HTTP GET (see 7.9.1) on this resource enumerates all
2359 instances of a given class in the namespace of this resource. The namespace is implied from
2360 this resource. The class is specified by the client using the \$class query parameter (see 6.5.1).
- 2361 • instance creation resource: A HTTP POST (see 7.5.1) on this resource creates an instance of a
2362 given class in the namespace of this resource (and thus the corresponding managed object).
2363 The namespace is implied from this resource. The class is specified by the client using the
2364 \$class query parameter (see 6.5.1).
- 2365 • method invocation resources for static methods: A HTTP POST (see 7.10.3) on such a resource
2366 invokes a static method on a class in a namespace. Class, method and namespace are implied
2367 from this resource, and are also specified in the server entry point resource.

2368 Clients need to know class and namespace of some entry point instance(s) of the model(s) they want to
2369 interact with, to get beyond this server entry point, and can use the instance enumeration resource to
2370 retrieve these instances.

2371 **7.12.1 ServerEntryPoint payload element**

2372 A ServerEntryPoint payload element is the representation of a server entry point resource in the protocol.

2373 A ServerEntryPoint payload element shall have the attributes defined in Table 14.

2374 **Table 14 – Attributes of a ServerEntryPoint payload element**

Attribute name	Payload datatype	Requirement	Description
kind	String	Mandatory	the kind of the payload element; shall have the value "serverentrypoint"
self	URI	Mandatory	resource identifier of the server entry point resource
namespaces	SEPNamespace []	Mandatory	unordered set of entities with information about CIM namespaces exposed by the server using the CIM-RS protocol, as described in Table 15
entitytagging	Boolean	Mandatory	indicates whether the entity tagging feature (see 7.4.1) is implemented by the server
defaultpaging timeout	Integer	Mandatory	indicates the default paging timeout of the server. For details on paged retrieval, see 7.3.8

Attribute name	Payload datatype	Requirement	Description
minpaging timeout	Integer	Mandatory	indicates the minimum value clients may specify with the \$pagingtimeout query parameter (see 6.5.7). For details on paged retrieval, see 7.3.8
maxpaging timeout	Integer	Mandatory	indicates the maximum value clients may specify with the \$pagingtimeout query parameter (see 6.5.7). For details on paged retrieval, see 7.3.8
continueonerror	Boolean	Mandatory	indicates whether or not the server supports continuation on error during paged retrieval. For details on paged retrieval, see 7.3.8

2375 Each entry in the "namespaces" array attribute shall have the child attributes defined in Table 15.

2376 **Table 15 – Attributes of SEPNamespace payload datatype**

Attribute name	Payload datatype	Requirement	Description
name	String	Mandatory	name of the namespace (e.g. "root/cimv2"). Note that because the namespace names are represented as strings, any slash characters in the namespace names shall not be percent-encoded as they would when used in resource identifiers (see 6.3).
enumeration	URI	Mandatory	resource identifier of the instance enumeration resource for this namespace (see 7.9)
creation	URI	Mandatory	resource identifier of the instance creation resource for this namespace (see 7.5)
staticmethods	MethodLink []	Mandatory	unordered set of method invocation links (see 7.2.1), for all implemented static methods for this namespace. Condition: The array element includes method invocation links
protocolversions	String []	Mandatory	unordered set of all CIM-RS protocol versions supported by this namespace. Each array entry shall be one protocol version string. Each protocol version string shall be of the format "m.n.u", where m is the major version, n is the minor version and u is the update version. Note that the draft level is not part of the version string. Each of these version indicator strings (that is, m, n, and u) shall be a decimal representation of the corresponding version indicator number without leading zeros. Note that version indicator numbers may have more than a single decimal digit
contenttypes	String []	Mandatory	unordered set of all CIM-RS payload representations supported by this namespace. Each array entry shall be the media type identifying a payload representation, including its version (see 9.1.2.1)

2377 **7.12.2 GET**

2378 **Purpose:** Retrieves the entry point resource of a server

2379 **HTTP method:** GET

2380 **Target resource:** Server entry point resource (see 7.12)

- 2381 **Query parameters:** None
- 2382 **Request headers:** Host, X-CIMRS-Version
- 2383 **Request payload:** None
- 2384 **Response headers (success):** Date, X-CIMRS-Version
- 2385 **Response payload (success):** ServerEntryPoint (see 7.12.1)
- 2386 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version
- 2387 **Response payload (failure):** ErrorResponse (see 7.3.6)
- 2388 **Requirement:** Mandatory
- 2389 **Description:**
- 2390 The HTTP GET method on a server entry point resource retrieves a representation of the specified
 2391 server entry point resource. The returned ServerEntryPoint payload element describes protocol-level
 2392 capabilities of the server and its namespaces, such as supported protocol versions and supported
 2393 payload representations, as well as resource identifiers for discovering further resources in the
 2394 server and its namespaces.
- 2395 On success, the entity body shall contain a ServerEntryPoint payload element (see 7.12.1) and one
 2396 of the following HTTP status codes shall be returned:
- 2397 • 200 "OK": The entity body contains the response payload element
 - 2398 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
 2399 empty. This status code can only occur if the server supports conditional requests and the
 2400 client has requested a conditional request
- 2401 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
 2402 the following HTTP status codes shall be returned:
- 2403 • 404 "Not Found": Target server entry point resource does not exist
 - 2404 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
 2405 method (see [RFC2616](#))

2406 **Example HTTP conversation:**

2407 **Request:**

```
2408 GET /cimrs HTTP/1.1
2409 Host: server.acme.com:5988
2410 Accept: application/json;version=1.0
2411 X-CIMRS-Version: 1.0.0
```

2412 **Response:**

```
2413 HTTP/1.1 200 OK
2414 Date: Fri, 11 Nov 2011 10:11:00 GMT
2415 Content-Length: XXX
2416 Content-Type: application/json;version=1.0.1
2417 X-CIMRS-Version: 1.0.1
2418
2419 {
```

```

2420 "kind": "serverentrypoint",
2421 "self": "/cimrs",
2422 "namespaces": [
2423   { "name": "interop",
2424     "enumeration": "/cimrs/interop/enum",
2425     "creation": "/cimrs/interop/create",
2426     "staticmethod": "/cimrs/interop/static",
2427     "protocolversions": [ "1.0.0", "1.0.1" ],
2428     "contenttypes": [
2429       "application/json;version=1.0.0",
2430       "application/json;version=1.0.1",
2431       "text/xml;version=1.0.0" ]
2432   },
2433   { "name": "root/cimv2",
2434     "enumeration": "/cimrs/root%2Fcimv2/enum",
2435     "creation": "/cimrs/root%2Fcimv2/create",
2436     "staticmethod": "/cimrs/root%2Fcimv2/static",
2437     "protocolversions": [ "1.0.0", "1.0.1" ],
2438     "contenttypes": [
2439       "application/json;version=1.0.0",
2440       "application/json;version=1.0.1",
2441       "text/xml;version=1.0.0" ]
2442   }
2443 ],
2444 "entitytagging": true,
2445 "pagedretrieval": true,
2446 "defaultpagingtimeout": 300,
2447 "minimumpagingtimeout": 1,
2448 "maximumpagingtimeout": 600,
2449 "continueonerror": true
2450 }

```

2451 **7.13 Listener entry point resource**

2452 A listener entry point resource describes protocol-level capabilities of a listener.

2453 **7.13.1 ListenerEntryPoint payload element**

2454 A ListenerEntryPoint payload element is the representation of a listener entry point resource.

2455 A ListenerEntryPoint payload element shall have the attributes defined in Table 16.

2456 **Table 16 – Attributes of a ListenerEntryPoint payload element**

Attribute name	Payload datatype	Requirement	Description
kind	String	Mandatory	the kind of the payload element; shall have the value "listenerentrypoint"
self	URI	Mandatory	resource identifier of the listener entry point resource
destinations	URI []	Mandatory	unordered set of resource identifiers of the listener destination resources of the listener (see 7.11)

Attribute name	Payload datatype	Requirement	Description
protocolversions	String []	Mandatory	unordered set of all CIM-RS protocol versions supported by the listener. Each array entry shall be one protocol version string. Each protocol version string shall be of the format "m.n.u", where m is the major version, n is the minor version and u is the update version. Note that the draft level is not part of the version string. Each of these version indicator strings (that is, m, n, and u) shall be a decimal representation of the corresponding version indicator number without leading zeros. Note that version indicator numbers may have more than a single decimal digit
contenttypes	String []	Mandatory	unordered set of all CIM-RS payload representations supported by the listener. Each array entry shall be the media type identifying a payload representation, including its version (see 9.1.2.1)

2457 7.13.2 GET

2458	Purpose:	Retrieves the entry point resource of a listener
2459	HTTP method:	GET
2460	Target resource:	Listener entry point resource (see 7.13)
2461	Query parameters:	None
2462	Request headers:	Host, X-CIMRS-Version
2463	Request payload:	None
2464	Response headers (success):	Date, X-CIMRS-Version
2465	Response payload (success):	ListenerEntryPoint (see 7.13.1)
2466	Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
2467	Response payload (failure):	ErrorResponse (see 7.3.6)
2468	Requirement:	Mandatory

2469 Description:

2470 The HTTP GET method on a listener entry point resource retrieves a representation of the specified
 2471 listener entry point resource. The returned ListenerEntryPoint payload element describes protocol-
 2472 level capabilities of a listener, such as supported protocol versions and supported payload
 2473 representations.

2474 On success, the entity body shall contain a ListenerEntryPoint payload element (see 7.13.1) and one
 2475 of the following HTTP status codes shall be returned:

- 2476 • 200 "OK": The entity body contains the response payload element
- 2477 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
 2478 empty. This status code can only occur if the server supports conditional requests and the
 2479 client has requested a conditional request

2480 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
 2481 the following HTTP status codes shall be returned:

- 2482 • 404 "Not Found": Target listener entry point resource does not exist
- 2483 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
- 2484 method (see [RFC2616](#))

2485 **Example HTTP conversation (server to listener):**

2486 Request:

```
2487 GET /cimrs HTTP/1.1
2488 Host: listener.acme.com:5988
2489 Accept: application/json;version=1.0
2490 X-CIMRS-Version: 1.0.1
```

2491 Response:

```
2492 HTTP/1.1 200 OK
2493 Date: Fri, 11 Nov 2011 10:11:00 GMT
2494 Content-Length: XXX
2495 Content-Type: application/json;version=1.0.0
2496 X-CIMRS-Version: 1.0.0
2497
2498 {
2499   "kind": "listenerentrypoint",
2500   "self": "/cimrs",
2501   "destinations": [ "/cimrs/dest1", "/cimrs/dest2" ],
2502   "protocolversions": [ "1.0.0" ],
2503   "contenttypes": [
2504     "application/json;version=1.0.0" ]
2505 }
```

2506 **7.14 CIM-RS resources to be exposed**

2507 This subclause summarizes which resources servers and listeners need to expose.

2508 **7.14.1 Resources exposed by a server**

2509 The following resources shall be exposed once by a server:

- 2510 • Server entry point resource (see 7.12)

2511 For each namespace that is supported for access by the CIM-RS protocol, the following resources shall
2512 be exposed by a server:

- 2513 • Instance enumeration resource (see 7.9)
- 2514 • Instance creation resource (see 7.5)
- 2515 • Method invocation resource (see 7.10) for static methods

2516 For each instance (including association instances) in each namespace that is supported for access by
2517 the CIM-RS protocol, the following resources shall be exposed by a server:

- 2518 • Instance resource (see 7.6)
- 2519 • Instance collection resources (see 7.8) and reference collection resources (see 7.7) that
- 2520 continue retrieval of such collections in paged mode. Note that the presence of these collections
- 2521 is highly dynamic

- 2522 • Method invocation resources (see 7.10); one for each non-static method that is exposed by the
2523 creation class of the instance and that is implemented

2524 **7.14.2 Resources exposed by a listener**

2525 The following resources shall be exposed once by a listener:

- 2526 • Listener entry point resource (see 7.13)

2527 For each listener destination supported by a listener, the following resources shall be exposed by the
2528 listener:

- 2529 • Listener destination resource (see 7.11)

2530 **7.15 Other typical WBEM protocol functionality**

2531 Certain functionality that is typical for a WBEM protocol or for systems management protocols in general
2532 does not have specific operations defined in the CIM-RS protocol, but can be performed by using other
2533 operations defined in the CIM-RS protocol, or discovery protocols, or the functionality of model-defined
2534 management interfaces accessible through the CIM-RS protocol. This subclause describes how a
2535 number of such functionalities can be performed.

2536 **7.15.1 Server discovery**

2537 WBEM servers can be discovered as described in clause 10.

2538 **7.15.2 Discovery of server and listener entry point resources**

2539 Once the IP address or hostname of a server or listener is known, the well-known resource identifier for
2540 its entry point resources can be constructed as described in 6.6, and using those, their entry point
2541 resources can be retrieved by performing the HTTP GET method on a server entry point resource (see
2542 7.12.2) and listener entry point resource (see 7.13.2), respectively.

2543 **7.15.3 Namespace discovery**

2544 The set of namespaces implemented by a server that support access through the CIM-RS protocol can
2545 be discovered from the "namespaces" attribute of the server entry point resource (see 7.12).

2546 **7.15.4 Registered profile discovery**

2547 The Profile Registration Profile ([DSP1033](#)) describes how to discover the management profiles to which a
2548 server advertises conformance, and from there, all further resources that are part of the functionality of a
2549 management profile. The management profiles to which a server advertises conformance can be
2550 discovered by enumerating instances of the CIM_RegisteredProfile class in the Interop namespace using
2551 the HTTP GET method on the instance enumeration resource for the Interop namespace (see 7.9.1).

2552 **7.15.5 Schema inspection**

2553 The schema definition (that is, class declarations and qualifier type declarations) including its meta-data
2554 in the form of qualifiers is expected to be accessible through a future "schema inspection model", using
2555 the existing operations defined in the CIM-RS protocol.

2556 **7.15.6 Association traversal**

2557 The CIM-RS protocol supports traversal of associations from a source instance to the association
2558 instances referencing the source instance, and to the instances associated with the source instance.
2559 There is no specific operation defined for this. Instead, it is performed by using the \$expand (see 6.5.3) or

2560 \$refer (see 6.5.9) query parameters to cause the inclusion of navigation properties for association
2561 traversal. For details on navigation properties, see 5.6.

2562 **7.15.7 Indication subscription**

2563 The CIM-RS protocol defines the HTTP POST method on listener destination resources (see 7.11.2) for
2564 the delivery of indications (that is, event notifications). However, it does not define any specific operations
2565 for performing other indication-related functions such as subscribing for indications, retrieving and
2566 managing indication filters and filter collections, or retrieving and managing listener destinations or
2567 indication services.

2568 Consistent with other WBEM protocols, the CIM-RS protocol leaves the definition of such functionality to a
2569 model-defined management interface, such as the *Indications Profile* ([DSP1054](#)).

2570 **8 HTTP usage**

2571 **8.1 General requirements**

2572 WBEM clients, servers, and listeners may support the use of HTTP for the CIM-RS protocol. The
2573 following applies if HTTP is supported:

- 2574 • Version 1.1 of HTTP shall be supported as defined in [RFC2616](#).
- 2575 • Version 1.0 or earlier of HTTP shall not be supported.

2576 WBEM clients, servers, and listeners shall support the use of HTTPS for the CIM-RS protocol. The
2577 following applies:

- 2578 • HTTPS shall be supported as defined in [RFC2818](#).
- 2579 • Within HTTPS, version 1.1 of HTTP shall be supported as defined in [RFC2616](#).

2580 NOTE 1 HTTPS should not be confused with Secure HTTP defined in RFC2660.

2581 **8.2 Authentication requirements**

2582 This subclause describes requirements and considerations for authentication between clients, servers,
2583 and listeners. Specifically, authentication happens from clients to servers for operation messages, and
2584 from servers to listeners for indication delivery messages.

2585 **8.2.1 Operating without authentication**

2586 WBEM clients, servers, and listeners may support operating without the use of authentication.

2587 This may be acceptable in environments such as physically isolated networks or between components on
2588 the same operating system.

2589 **8.2.2 HTTP basic authentication**

2590 HTTP basic authentication provides a rudimentary level of authentication, with the major weakness that
2591 the client password is part of the HTTP headers in unencrypted form.

2592 WBEM clients, servers, and listeners may support HTTP basic authentication as defined in [RFC2617](#).

2593 HTTP basic authentication may be acceptable in environments such as physically isolated networks,
2594 between components on the same operating system, or when the messages are encrypted by using
2595 HTTPS.

2596 8.2.3 HTTP digest authentication

2597 HTTP digest authentication verifies that both parties share a common secret without having to send that
2598 secret in the clear. Thus, it is more secure than HTTP basic authentication.

2599 WBEM clients, servers, and listeners should support HTTP digest authentication as defined in [RFC2617](#).

2600 8.2.4 Other authentication mechanisms

2601 WBEM clients, servers, and listeners may support authentication mechanisms not covered by [RFC2617](#).
2602 One example of such a mechanism is public key certificates as defined in [X.509](#).

2603 8.3 Message encryption requirements

2604 Encryption of HTTP messages can be supported by the use of HTTPS and its secure sockets layer.

2605 It is important to understand that authentication and encryption of messages are separate issues:
2606 Encryption of messages requires the use of HTTPS, while the authentication mechanisms defined in 8.2
2607 can be used with both HTTP and HTTPS.

2608 The following requirements apply to clients, servers, and listeners regarding the secure sockets layer
2609 used with HTTPS:

- 2610 • TLS 1.0 (also known as SSL 3.1) as defined in [RFC2246](#) shall be supported. Note that TLS 1.0
2611 implementations may be vulnerable when using CBC cipher suites
- 2612 • TLS 1.1 as defined in [RFC4346](#) should be supported
- 2613 • TLS 1.2 as defined in [RFC5246](#) should be supported
- 2614 • SSL 2.0 or SSL 3.0 shall not be supported because of known security issues in these versions

2615 Note that given these requirements, it is valid to support only TLS 1.0 and TLS 1.2 but not TLS 1.1. At the
2616 time of publication of this standard, it is expected that support for TLS 1.1 and TLS 1.2 is still not
2617 pervasive; therefore TLS 1.0 has been chosen as a minimum despite its known security issues.

2618 [RFC5246](#) describes in Appendix E "Backward Compatibility" how the secure sockets layer can be
2619 negotiated.

2620 The following requirements apply to clients, servers, and listeners regarding the cipher suites used with
2621 HTTPS:

- 2622 • The TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value 0x0013)
2623 shall be supported when using TLS 1.0. Note that [RFC2246](#) defines this cipher suite to be
2624 mandatory for TLS 1.0
- 2625 • The TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value 0x000A) shall
2626 be supported when using TLS 1.1. Note that [RFC4346](#) defines this cipher suite to be mandatory
2627 for TLS 1.1
- 2628 • The TLS_RSA_WITH_AES_128_CBC_SHA cipher suite (hexadecimal value 0x002F) shall be
2629 supported when using TLS 1.2. Note that [RFC5246](#) defines this cipher suite to be mandatory for
2630 TLS 1.2
- 2631 • The TLS_RSA_WITH_AES_128_CBC_SHA256 cipher suite (hexadecimal value 0x003C)
2632 should be supported when using TLS 1.2, in order to meet the transition to a security strength of
2633 112 bits (guidance is provided in NIST Special Publication 800-57 [\[NIST 800-57\]](#) and NIST
2634 Special Publication 800-131A [\[NIST 800-131A\]](#))
- 2635 • Any additional cipher suites may be supported

2636 8.4 HTTP header fields

2637 This subclause describes the use of HTTP header fields within the CIM-RS protocol, and it defines
2638 extension-header fields specific to the CIM-RS protocol.

2639 Any rules for processing header fields defined in [RFC2616](#) apply, particularly regarding whitespace
2640 stripping, line continuation, multiple occurrences of headers, and case insensitive treatment of field
2641 names.

2642 8.4.1 Accept

2643 The rules for the Accept request-header field defined in [RFC2616](#) apply. This subclause defines
2644 additional constraints on its use.

2645 The Accept header field may be provided on the request message of any operation that may return a
2646 response payload.

2647 If provided by a client, the Accept header field shall specify media types identifying CIM-RS payload
2648 representations (including version) that are supported by the client.

2649 The use of media ranges (that is, the asterisk character "*") in the type or subtype fields of the media type
2650 is not permitted in the CIM-RS protocol.

2651 NOTE: [RFC2616](#) permits the use of media ranges for the Accept header field. However, with the envisioned
2652 combinations of type and subtype values for CIM-RS, wildcarding based on type and subtype is not meaningful.

2653 If implemented, the "q" accept parameter shall be interpreted as a preference; interpreting it as a quality
2654 does not make sense for the CIM-RS protocol. Clients may provide the "q" accept parameter. Servers
2655 should implement the "q" accept parameter; if not implemented, it shall be tolerated if provided.

2656 NOTE: [RFC2616](#) does not specify recommendations for implementing the "q" accept parameter.

2657 NOTE: [RFC2616](#) distinguishes between general media type parameters (such as "version"), and accept
2658 parameters (such as "q"); the latter can be used only in the Accept header field, while general media type parameters
2659 can be considered part of the media type definition.

2660 Additional accept parameters (that is, beyond "q") are not permitted to be used in the Accept header field.
2661 For future extensibility, servers shall tolerate and ignore unknown additional accept parameters.

2662 If an Accept header field is provided, servers shall use one of the payload representations and version
2663 identified in the Accept header field for the response payload, considering the "q" accept parameter if
2664 implemented.

2665 The version specified in the "version" parameter of a media type shall be interpreted by the server as
2666 follows:

- 2667 • If an update version is included, it specifies the lowest acceptable update version (within the
2668 specified major version and acceptable minor versions); higher update versions shall be
2669 acceptable in addition. If no update version is included, the server shall assume a default of 0;
2670 that is, any update version is acceptable (within the specified major version and acceptable
2671 minor versions).
- 2672 • The minor version specifies the only acceptable minor version.
- 2673 • The major version specifies the only acceptable minor version.

2674 NOTE: These rules follow the usual DMTF convention for referencing versions: Update versions newer than the
2675 one specified are selected automatically if available, but newer minor (and of course, major) versions are selected
2676 automatically.

2677 If none of the payload representations identified in the Accept header field is supported by the server, it
2678 shall return HTTP status code 406 "not acceptable".

2679 NOTE: [RFC2616](#) only recommends returning HTTP status code 406 "not acceptable" in this case, but it does not
2680 require it.

2681 If no Accept header field is provided, servers may use any valid payload representation and version for
2682 the response payload.

2683 Within the constraints defined in this subclause, the payload representations specified in the Accept
2684 header field and the payload representations used in the response may change over time, even between
2685 the same combination of client and server. This implies that a server needs to evaluate the Accept header
2686 field (if present) on every request, even when the request is originated from the same client as before.

2687 Example:

```
2688 Accept: application/json; version=2.0,  
2689         application/json;version=1.0.1; q=0.5,  
2690         text/xml; version=1.0;q=0.2
```

2691 In this example, value of the Accept header field is distributed over multiple lines. The client
2692 expresses a preference for version 2.0.x (x>=0) of the CIM-RS JSON payload representation (by
2693 means of the default value of 1 for the "q" parameter), if that representation version is not available,
2694 then for version 1.0.x (x>=1) of the CIM-RS JSON representation, if that is not available then for
2695 version 1.0.x (x>=0) of the CIM-RS XML representation.

2696 8.4.2 Content-Type

2697 The rules for the Content-Type entity-header field defined in [RFC2616](#) apply. This subclause defines
2698 additional constraints on its use.

2699 As defined in [RFC2616](#), the Content-Type entity-header field shall be provided on the request message
2700 of any operation that passes a request payload and on the response message of any operation that
2701 returns a response payload.

2702 The Content-Type entity-header field shall specify the media type identifying the CIM-RS payload
2703 representation and version that is used for the content of the entity body. The "version" parameter of the
2704 media type shall include the major, minor and update version indicators.

2705 8.4.3 ETag (EXPERIMENTAL)

2706 EXPERIMENTAL

2707 The rules for the ETag response-header field defined in [RFC2616](#) apply. This subclause defines
2708 additional constraints on its use.

2709 The ETag response-header field shall be provided in the response to a HTTP GET method on an
2710 instance resource (see 7.6.3), if the entity tagging feature (see 7.4.1) is implemented by the server.

2711 In this case, the ETag response-header field shall be specified using the following format (defined in
2712 ABNF):

```
2713 ETag = "ETag" WS ":" entity-tag
```

2714 where `entity-tag` is a suitable entity tag as defined in [RFC2616](#), and `WS` is whitespace as defined in
2715 subclause "ABNF usage conventions". In models based on the CIM Schema published by DMTF, the
2716 Generation property defined in class `CIM_ManagedElement` is targeted for that purpose.

2717 Otherwise, the ETag response-header field shall not be provided by a server.

2718 The ETag response-header field shall not be provided in any other responses.

2719 EXPERIMENTAL

2720 8.4.4 If-Match (EXPERIMENTAL)

2721 EXPERIMENTAL

2722 The rules for the If-Match request-header field defined in [RFC2616](#) apply. This subclause defines
2723 additional constraints on its use.

2724 The If-Match request-header field may be provided in the request of a HTTP PUT method on an instance
2725 resource (see 7.6.4), if the entity tagging feature (see 7.4.1) is implemented by the client and the server
2726 that returned the instance that is being modified, has implemented the entity tagging feature as well.

2727 If provided, the If-Match request-header field shall be specified using the following format for its field value
2728 (defined in ABNF):

```
2729 If-Match-value = entity-tag
```

2730 where `entity-tag` is the entity tag of the ETag header field of the retrieved representation of the
2731 instance resource that is the basis for the modification.

2732 The If-Match request-header field shall not be provided in any other requests.

2733 EXPERIMENTAL

2734 8.4.5 X-CIMRS-Version

2735 The CIM-RS protocol version is the version of this document, without any draft level. The X-CIMRS-
2736 Version extension-header field shall identify the CIM-RS protocol version to which the request or
2737 response conforms, using the following format for its field value (defined in ABNF):

```
2738 X-CIMRS-Version-value = M "." N "." U
```

2739 where `M` is the major version indicator, `N` is the minor version indicator, and `U` is the update version
2740 indicator within the version. Each of these version indicator strings shall be a decimal representation of
2741 the corresponding version indicator number without leading zeros. Note that each indicator version string
2742 may include more than a single decimal digit.

2743 The X-CIMRS-Version extension-header field shall be included in any request and in any response.

2744 Example:

```
2745 X-CIMRS-Version: 1.0.0
```

2746 9 Payload representation

2747 CIM-RS payload representation specifications define how the abstract payload elements defined in this
2748 document are encoded in the entity body of the HTTP messages used by the CIM-RS protocol. Such an
2749 encoding format is termed a "payload representation" in this document.

2750 This clause defines requirements for payload representation specifications and for implementations of the
2751 CIM-RS protocol that are related to payload representations.

2752 9.1 Internet media types

2753 The CIM-RS protocol uses Internet media types, as defined in section 3.7 of [RFC2616](#), for identifying the
2754 payload representation of its abstract payload elements. This subclause defines requirements related to
2755 media types used for the CIM-RS protocol.

2756 9.1.1 General

2757 CIM-RS payload representation specifications shall define a single media type that uniquely identifies a
2758 payload representation across all payload representations listed in Table 18.

2759 It is recommended that any such media types be registered with IANA.

2760 Any media types used for the CIM-RS protocol shall identify the version of the payload representation
2761 using a media type parameter named "version", as described in 9.1.2.1.

2762 Example of a media type that is valid for the CIM-RS protocol:

2763 `application/json; version=1.0`

2764 9.1.2 Media type parameters

2765 Table 17 defines parameters of media types used for the CIM-RS protocol. Parameters not listed in the
2766 table are not permitted to be used. For future extensibility, consumers of media types shall tolerate and
2767 ignore unknown media type parameters.

2768 **Table 17 – Media type parameters**

Parameter	Presence Requirement	Description
version	Mandatory	See 9.1.2.1.

2769 9.1.2.1 Parameter "version"

2770 The media type parameter named "version" shall identify the version of the payload representation
2771 identified by the media type, using the following format for its value (defined in ABNF):

2772 `version-value = M ["." N ["." U]]`

2773 where *M* is the major version indicator, *N* is the minor version indicator, and *U* is the update version
2774 indicator within the version. Each of these version indicator strings shall be a decimal representation of
2775 the corresponding version indicator number without leading zeros. Note that each indicator version string
2776 may include more than a single decimal digit.

2777 Subclauses in this document that describe the usage of media types define additional requirements on
2778 the presence of the minor and update version indicators in the value of the "version" parameter.

2779 The semantics for these version indicators shall be the semantics defined by DMTF for its specification
 2780 versions. The version indicators of payload representation specifications provided by third parties shall
 2781 conform to that semantics.

2782 9.2 Payload element representations

2783 CIM-RS payload representation specifications shall define a representation for each payload element
 2784 listed in Table 4.

2785 The representations of these payload elements should be designed such that they can represent
 2786 elements from any valid model without introducing restrictions, and such that there is no need to extend
 2787 the payload representation specification if the model gets extended.

2788 Attributes of the payload elements defined in this document may be represented in any way in the
 2789 payload representation. The attribute names stated in the descriptions of the payload elements in clause
 2790 7 do not need to be retained in the payload representation. The payload datatypes stated in Table 5 do
 2791 not need to correspond 1:1 to datatypes the representation format may use, as long as the value range of
 2792 the attribute values can be correctly represented without any restrictions or loss of information.

2793 For example, in a JSON representation of an Instance payload element (see 7.6.1), all of the following
 2794 options would be valid for representing the "self" attribute for resource identifier "/cimrs/machine/1234":

- 2795 • as a JSON attribute with the same name as the attribute of the abstract payload element:

```
2796 {
2797   "self": "/cimrs/machine/1234",
2798   . . .
2799 }
```

- 2800 • as a JSON attribute with a different name as the attribute of the abstract payload element:

```
2801 {
2802   "this": "/cimrs/machine/1234",
2803   . . .
2804 }
```

- 2805 • as an entry in a JSON array for links following the rel/href approach:

```
2806 {
2807   "links": [
2808     { "rel": "self",
2809       "href": "/cimrs/machine/1234" },
2810     . . .
2811   ],
2812   . . .
2813 }
```

2814 9.3 Payload representations

2815 Table 18 lists known payload representations and requirements to implement them; payload
 2816 representations not listed in Table 18 may be implemented in addition.

2817 This table will be kept up to date in future versions of this document to include known payload
 2818 representations, in order to provide a basis on which the media type can be kept unique.

2819

Table 18 – CIM-RS payload representations

Name	Requirement	Underlying format	Defined in
CIM-RS Payload Representation in JSON	Mandatory	JavaScript Object Notation (JSON)	DSP0211

2820

2821 10 Discovery requirements

2822 The CIM-RS protocol has the following requirements related to discovery protocols:

2823 Wbem servers should implement the SLP discovery protocol, supporting the provisions set forth in
2824 [DSP0205](#), supporting the SLP template defined in [DSP0206](#).

2825 The CIM-RS protocol has no requirements for supporting the discovery of listeners. Note that listeners are
2826 HTTP servers.

2827 11 Version compatibility

2828 This clause defines the rules for version compatibility between Wbem clients and servers.

2829 Since HTTP is session-less, the general principle for determining version compatibility in the CIM-RS
2830 protocol is that the version for the relevant layers of the CIM-RS protocol is included in all protocol
2831 messages, allowing the receiving participant to determine whether it is able to support that version.

2832 The general principle for backwards compatibility (as further detailed in this clause) is that servers are
2833 backwards compatible to clients; that is, servers of a particular version work with "older" versions of
2834 clients.

2835 Version compatibility for the CIM-RS protocol is defined for the following protocol layers:

- 2836 • HTTP protocol (see 11.1)
- 2837 • CIM-RS protocol (see 11.2)
- 2838 • CIM-RS payload representation (see 11.3)

2839 A client and a server are version-compatible with each other only if they are compatible at each of these
2840 three protocol layers.

2841 11.1 HTTP protocol version compatibility

2842 As defined in [RFC2616](#), every HTTP request and every HTTP response shall indicate the HTTP protocol
2843 version to which the message format conforms.

2844 Since the CIM-RS protocol requires support for HTTP 1.1 (see 8.1), the backward compatibility rules for
2845 supporting HTTP 1.0 and HTTP 0.9 as defined in section 19.6 (Compatibility with Previous Versions) of
2846 [RFC2616](#) do not need to be followed in order to conform to the CIM-RS protocol.

2847 At this point, there is no HTTP version higher than 1.1 defined. Therefore, a client and a server are
2848 compatible w.r.t. the HTTP protocol version only if they both support HTTP 1.1.

2849 11.2 CIM-RS protocol version compatibility

2850 As defined in 8.4.5, every HTTP request and every HTTP response in the CIM-RS protocol shall indicate
2851 the CIM-RS protocol version to which the request or response conforms, by including the X-CIMRS-

2852 Version extension-header field. As defined in 8.4.5, the X-CIMRS-Version extension-header field
2853 identifies major, minor and update version of the CIM-RS protocol.

2854 A client and a server are compatible w.r.t. the CIM-RS protocol version only if the following condition is
2855 satisfied:

- 2856 • the major version of the server is equal to the major version of the client, and the minor version
2857 of the server is equal to or larger than the minor version of the client.

2858 The update version is not considered in this rule because new update versions (within the same major
2859 and minor version) are not supposed to introduce new functionality, so this rule allows clients and servers
2860 to be upgraded to conform to new update versions of the CIM-RS protocol independently of each other.

2861 **11.3 CIM-RS payload representation version compatibility**

2862 As defined in 9.1, the CIM-RS payload representation is identified using a media type whose "version"
2863 parameter identifies its major, minor and update version.

2864 A client and a server are compatible w.r.t. the version of a particular payload representation only if the
2865 following condition is satisfied:

- 2866 • the major version of the server is equal to the major version of the client, and the minor version
2867 of the server is equal to or larger than the minor version of the client.

2868 The update version is not considered in this rule because new update versions (within the same major
2869 and minor version) are not supposed to introduce new functionality, so this rule allows clients and servers
2870 to be upgraded to conform to new update versions of the payload representation independently of each
2871 other.

2872 **12 Conformance**

2873 This clause defines the criteria for WBEM clients, servers, and listeners to implement the CIM-RS
2874 protocol conformant to this document.

2875 WBEM clients, servers, and listeners implement the CIM-RS protocol conformant to this document only if
2876 they satisfy all provisions set out in this document.

2877 The terms client, server, and listener in this document refer to clients, servers, and listeners that are
2878 conformant to this document, without explicitly mentioning that.

2879 **ANNEX A**
2880 (normative)

2881
2882 **Common ABNF rules**

2883 This annex defines common ABNF rules used throughout this document.

2884 `nonZeroDecimalDigit = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"`

2885 `decimalDigit = "0" / nonZeroDecimalDigit`

2886 `leadingZeros = 1*"0"`

2887 `positiveDecimalInteger = [leadingZeros] nonZeroDecimalDigit *decimalDigit`

2888 `nonNegativeDecimalInteger = [leadingZeros] ("0" / nonZeroDecimalDigit *decimalDigit)`

2889

ANNEX B (informative)

2890
2891
2892
2893

Mapping CIM-RS to generic operations

2894 This annex describes how CIM-RS is to be mapped to generic operations (see [DSP0223](#)). This mapping
2895 can be used when adding support for the CIM-RS protocol to CIM servers that internally support the
2896 semantics of generic operations either directly or indirectly through a (further) mapping.

2897 **B.1 URI composition**

2898 CIM-RS does not specify the structure of URIs. URIs are considered opaque to the client, leaving each
2899 server implementation free to structure them as necessary. However, there will be some units of
2900 information that the server must be able to infer from a particular URI, and be able to perform bidirectional
2901 lossless translations between the URI and the information units. The server is free to enable this
2902 translation as it sees fit. This might be done by encoding the information into the URI, or by keeping a
2903 cache of the information indexed by a short hash that is encoded into the URI, or by any other means.

2904 The subclauses below describe the units of information that must be represented in the URI of each
2905 resource type (see Table 2). Unless otherwise stated, units of information are represented in the path
2906 component of the URI, in a server-specific way. Some information units are represented in CIM-RS query
2907 parameters, so they should not additionally be represented in the path component. Note that query
2908 parameters in a URI are considered part of the resource address (see [RFC3986](#)).

2909 **B.1.1 Instance creation resource**

2910 This resource represents the ability to create instance resources in a particular CIM namespace (see 7.5).
2911 Its URI enables the server to identify:

- 2912 • CIM namespace in which the new instance is to be created;
- 2913 • The name of the creation class of the instance to be created (represented in the URI through
2914 the \$class query parameter, see 6.5.1);
- 2915 • The type of the resource (in this case, an instance creation resource).

2916 **B.1.2 Instance resource**

2917 This resource represents a managed object in the managed environment, through a CIM instance (see
2918 7.6). Its URI enables the server to identify:

- 2919 • CIM namespace of the instance (this is also the namespace of its creation class);
- 2920 • Name of instance's creation class;
- 2921 • Key bindings of the instance (name/value pairs of all key properties);
- 2922 • The type of the resource (in this case, an instance resource).

2923 **B.1.3 Page of instance or reference collection resource from association traversal**

2924 An instance collection resource represents a collection of instance resources (see 7.8). A reference
2925 collection resource represents a collection of references to instance resources (see 7.7). Instance or
2926 reference collection resources representing the result of an association traversal from a source instance
2927 do not have URIs; their representation is always embedded as the value of a navigation property (see
2928 5.6) in the source instance. If such an instance or reference collection is returned using paging (see

2929 7.3.8), the pages following the initial (embedded) part of the collection have URIs. The URI of such a
2930 page enables the server to identify:

- 2931 • CIM namespace of the source instance;
- 2932 • Name of creation class of the source instance;
- 2933 • Key bindings of the source instance (name/value pairs of all key properties);
- 2934 • The relationship of the source instance to the result, represented in the URI through the
2935 `$expand` (see 6.5.3) and `$refer` (see 6.5.9) query parameters;
- 2936 • Some information identifying the page in the overall result;
- 2937 • The type of the resource and kind of result (in this case, a page of an instance or reference
2938 collection resource resulting from association traversal).

2939 **B.1.4 Page of instance or reference collection resource from enumeration by class**

2940 An instance collection resource represents a collection of instance resources (see 7.8). A reference
2941 collection resource represents a collection of references to instance resources (see 7.7). Instance or
2942 reference collection resources representing the result of an enumeration of instances of a given class do
2943 not have URIs; their representation is returned in the protocol payload (see 7.9). If such an instance or
2944 reference collection is returned using paging (see 7.3.8), the pages following the initial (payload) part of
2945 the collection have URIs. The URI of such a page enables the server to identify:

- 2946 • CIM namespace of the given class and the instances in the result set;
- 2947 • Name of the given class;
- 2948 • Some information identifying the page in the overall result;
- 2949 • The type of the resource and kind of result (in this case, a page of an instance or reference
2950 collection resource resulting from enumeration by class).

2951 **B.1.5 Instance enumeration resource**

2952 This resource represents the ability to enumerate instances of a given class (including instances of
2953 subclasses) in a particular CIM namespace (see 7.9). Its URI enables the server to identify:

- 2954 • CIM namespace of the given class;
- 2955 • Name of the given class (represented in the URI through the `$class` query parameter, see
2956 6.5.1);
- 2957 • The type of the resource (in this case, an instance enumeration resource).

2958 **B.1.6 Static method invocation resource**

2959 This resource represents the ability to invoke a static method upon a class that exposes that method (see
2960 7.10). Its URI enables the server to identify:

- 2961 • CIM namespace of the class upon which the method is to be invoked;
- 2962 • Name of the class upon which the method is to be invoked;
- 2963 • Name of the method;
- 2964 • The type of the resource (in this case, a static method invocation resource).

2965 **B.1.7 Non-static method invocation resource**

2966 This resource represents the ability to invoke a non-static method upon an instance whose creation class
2967 exposes that method (see 7.10). Its URI enables the server to identify:

- 2968 • CIM namespace of the instance upon which the method is to be invoked;
- 2969 • Name of the creation class of the instance upon which the method is to be invoked;
- 2970 • Key bindings of the instance upon which the method is to be invoked (name/value pairs of all
2971 key properties);
- 2972 • Name of the method;
- 2973 • The type of the resource (in this case, a non-static method invocation resource).

2974 **B.1.8 Listener destination resource**

2975 This resource represents the ability to deliver an indication to a listener (see 7.11). Its URI enables the
2976 server to identify:

- 2977 • The listener to which the indication is to be delivered;
- 2978 • The type of the resource (in this case, a listener destination resource).

2979 **B.1.9 Server and listener entry point resources**

2980 This resource describes protocol-level capabilities of a server or listener, and provides a starting point for
2981 discovering further resources in the server. This is the only resource for which CIM-RS specifies the
2982 format of the resource. Its URI encodes the following information:

- 2983 • The type of the resource (in this case, the server or listener entry point resource); this is
2984 specified to be: `/cimrs`

2985 **B.2 Query parameters**

2986 Specific query parameters can be used with multiple CIM-RS operation/resource pairs. Likewise, many
2987 input parameters are common between multiple generic operations, and are used consistently across
2988 those operations. With minor exceptions, the usage of any particular CIM-RS query parameter can be
2989 mapped directly to specific generic operation parameters, regardless of the CIM-RS operation/resource
2990 pair with which it is used.

2991 Table B-1 defines the mapping of CIM-RS query parameters to generic operations input parameters.

2992

Table B-1 – Mapping of CIM-RS query parameters to generic operations input parameters

CIM-RS Query Parameter	Generic Operations Input Parameter	Mapping
\$class		See individual operation/resource mappings in this annex
\$continueonerror	ContinueOnError	Directly equivalent
\$expand		See B.2.1
\$max	MaxObjectCount	Directly equivalent
\$methods	no equivalent	The \$methods query parameter has no analog in generic operations because it only dictates what links will be included in the returned payload. Logic to implement the \$methods query parameter will be confined to the server implementation's protocol handler and will not need to be passed on to providers or other server components.
\$pagingtimeout	OperationTimeout	Directly equivalent
\$properties	IncludedProperties and ExcludeSubclassProperties	\$properties is set to contents of IncludedProperties; if ExcludeSubclassProperties is TRUE, list of properties is reduced by those defined in subclasses.
\$refer		See B.2.1
\$filter	FilterQueryString and FilterQueryLanguage	Directly equivalent. If \$filter is specified, FilterQueryString is set to the \$filter query parameter value; FilterQueryLanguage is set to "DMTF:FQL" (see C.2)

2993 **B.2.1 Special handling for \$expand and \$refer query parameters**

2994 \$expand and \$refer direct the server to traverse associations or reference properties in the result set.

2995 Each \$expand or \$refer specification indicates one association traversal path, composed of an
2996 arbitrary number of association hops. Multiple paths may be specified in a single CIM-RS operation.

2997 \$expand and \$refer are permitted on CIM-RS operations which target a single instance or an instance
2998 collection. For each single instance, or each instance in a collection targeted by the CIM-RS operation,
2999 the server is directed to apply all \$expand and \$refer paths, thereby including the additional
3000 information requested.

3001 The values supplied to \$expand and \$refer query parameters are formatted in the same way. For
3002 either query parameter, the query parameter value is an association traversal path composed of an
3003 arbitrary length sequence of alternating association classes and reference properties, delimited by the
3004 period ('.') character. Each reference property within the path may have an optional class name to act as
3005 a filter on the types of instances to be considered at that point in the association traversal. Likewise for
3006 either query parameter, the association traversal path is applied to each instance targeted by the CIM-RS
3007 operation, and a representation of the final element in that traversal path is added to the result set.

3008 The difference between \$expand and \$refer is in the representation of the returned element. In the
3009 case of \$expand, the information returned is an instance collection representation of the terminal

3010 navigation hop element. In the case of `$refer`, the information returned is a reference collection of the
 3011 terminal navigation hop element.

3012 An implementation may do the following.

- 3013 1) Identify all association traversal paths identified in all `$expand` and `$refer` query parameters
 3014 supplied to the current operation. Merge the paths into a tree representation, so that common
 3015 early portions of the different traversal paths need not be redundantly traversed. In this way the
 3016 instance targeted by the CIM-RS operation is applied to the root of the traversal tree, and the
 3017 leaves of the traversal tree represent the results of the individual association traversal paths.
 3018 Note that if some traversal paths are strict supersets of others, this will result in a situation
 3019 where not all traversal paths end in leaf nodes of the traversal tree. For each instance targeted
 3020 by the CIM-RS operation, the tree is traversed to identify and supply the additional information
 3021 requested in the query parameters, as described in subsequent steps.
- 3022 2) When `$expand` or `$refer` is supplied for any CIM-RS operation, it will map to generic
 3023 operations in a common fashion regardless of which CIM-RS operation was invoked. In any
 3024 case, it is assumed that the CIM-RS operation being invoked will begin by obtaining an initial
 3025 instance or instance collection. Once that instance or collection is obtained, the following
 3026 generic operations mapping will be performed, using the initial instance or instance collection as
 3027 the "working instance collection".
- 3028 3) Obtain the initial association traversal element from the root of the traversal tree identified in
 3029 step 1) above.
- 3030 4) For each Working Instance in the working instance collection, perform the following. If the
 3031 current traversal tree node specifies both association class and reference, then perform a
 3032 generic operations `OpenAssociatedInstancePaths` operation; if only association class is
 3033 given, perform a generic operations `OpenReferencingInstancePaths` operation. (See step
 3034 6) below for possible modifications to generic operations method being called.) In either case,
 3035 the call is made with the following parameters:
- 3036 • `SourceInstancePath` is formed from:
 - 3037 – The CIM namespace (extracted from the Working Instance);
 - 3038 – The class name (extracted from the Working Instance);
 - 3039 – Key property name/value pairs (extracted from the Working Instance).
 - 3040 • `AssociationClassName` is extracted from the class name specified in the current
 3041 traversal tree node.
 - 3042 • `AssociatedClassName` is set to NULL.
 - 3043 • `SourceRoleName` is set to NULL.
 - 3044 • `AssociatedRoleName` is set to the reference name obtained from the current traversal
 3045 tree node, if reference name is present; if not present, `AssociatedRoleName` is set to
 3046 NULL.
 - 3047 • `FilterQueryString` is set from the `$filter` query parameter as described in B.2.1.
 - 3048 • `FilterQueryLanguage` is set to "DMTF:FQL" (see C.2).
 - 3049 • `OperationTimeout` is set from the `$pagingtimeout` query parameter as described in
 3050 Table B-1.
 - 3051 • `ContinueOnError` is set from the `$continueonerror` query parameter as described in
 3052 Table B-1.
 - 3053 • `MaxObjectCount` is set from the `$max` query parameter as described in Table B-1.

- 3054 5) If the current traversal tree node contains sub-nodes, then perform N recursions into step 4)
3055 above, setting the “current traversal tree node” to each of the N traversal tree sub-nodes.
- 3056 6) Special case: if the current traversal tree node corresponds to a terminal node in a `$expand`
3057 query parameter, then entire instances must be obtained instead of only instance paths.
3058 Therefore:
- 3059 a) Call `OpenAssociatedInstancesWithPath` instead of
3060 `OpenAssociatedInstancePaths`, or
- 3061 b) Call `OpenReferencingInstancesWithPath` operation instead of
3062 `OpenReferencingInstancePaths`.
- 3063 c) In either case, the following parameters will be supplied to the generic operations method:
- 3064 • `IncludeClassOrigin` is set to `FALSE`.
- 3065 • `IncludedProperties` is set from the `$properties` query parameter as described
3066 in Table B-1.
- 3067 • `ExcludeSubclassProperties` is set to `FALSE`.

3068 B.3 Server operations

3069 This subclause describes a server’s decision tree for how incoming CIM-RS operations are to be
3070 analyzed, identified, and mapped to generic operations: for each HTTP method, the server will examine
3071 its target URI. Based upon the server’s defined URI structure, it will determine what type of resource is
3072 targeted, and will then determine which generic operations are to be invoked.

3073 The following subclauses describe each combination of HTTP method and resource type (and in some
3074 cases, multiple variants of the same resource type).

3075 B.3.1 POST instance creation resource

3076 This CIM-RS operation creates an instance resource (see 7.5.1).

3077 This CIM-RS operation directly maps to the generic operation `CreateInstance`.

3078 The input parameters for this generic operation are formed as follows:

- 3079 • the `ClassPath` parameter is formed from:
- 3080 – the CIM namespace, which is formed from information units extracted from the target URI
3081 of the HTTP request (see B.1.1)
- 3082 – the class name, obtained from the `$class` query parameter in the target URI of the HTTP
3083 request (see B.1.1)
- 3084 • the `InstanceSpecification` parameter is formed from the class name and from the
3085 `properties` attribute of the `Instance` payload element in the HTTP request (see 7.6.1)

3086 The output parameters of this generic operation are used as follows:

- 3087 • the `InstancePath` parameter is used to form the URI in the `Location` header of the HTTP
3088 response

3089 Restrictions: None.

3090 **B.3.2 POST static method invocation resource**

3091 This CIM-RS operation invokes a static method defined in a class (extrinsic method), upon a class (see
3092 7.10.3).

3093 This CIM-RS operation directly maps to the generic operation `InvokeStaticMethod`.

3094 The input parameters for this generic operation are formed as follows:

- 3095 • the `ClassPath` parameter is formed from CIM namespace and class name, which are formed
3096 from information units extracted from the target URI of the HTTP request (see B.1.6)
- 3097 • the `MethodName` parameter is formed from information units extracted from the target URI of
3098 the HTTP request (see B.1.6)
- 3099 • the `InParmValues` parameter is formed from the `parameters` attribute of the
3100 `MethodRequest` payload element in the HTTP request (see 7.10.1)

3101 The output parameters of this generic operation are used as follows:

- 3102 • the `OutParmValues` parameter is used to form the `parameters` attribute of the
3103 `MethodResponse` payload element in the HTTP response (see 7.10.2)
- 3104 • the `ReturnValue` parameter is used to form the `returnvalue` attribute of the
3105 `MethodResponse` payload element in the HTTP response (see 7.10.2)

3106 Restrictions: None.

3107 **B.3.3 POST non-static method invocation resource**

3108 This CIM-RS operation invokes a non-static method defined in a class (extrinsic method), upon an
3109 instance (see 7.10.3).

3110 This CIM-RS operation directly maps to the generic operation `InvokeMethod`.

3111 The input parameters for this generic operation are formed as follows:

- 3112 • the `InstancePath` parameter is formed from CIM namespace, class name and key bindings,
3113 which are all formed from information units extracted from the target URI of the HTTP request
3114 (see B.1.7)
- 3115 • the `MethodName` parameter is formed from information units extracted from the target URI of
3116 the HTTP request (see B.1.7)
- 3117 • the `InParmValues` parameter is formed from the `parameters` attribute of the
3118 `MethodRequest` payload element in the HTTP request (see 7.10.1)

3119 The output parameters of this generic operation are used as follows:

- 3120 • the `OutParmValues` parameter is used to form the `parameters` attribute of the
3121 `MethodResponse` payload element in the HTTP response (see 7.10.2)
- 3122 • the `ReturnValue` parameter is used to form the `returnvalue` attribute of the
3123 `MethodResponse` payload element in the HTTP response (see 7.10.2)

3124 Restrictions: None.

3125 **B.3.4 DELETE instance resource**

3126 This CIM-RS operation deletes an instance resource (see 0).

3127 This CIM-RS operation directly maps to the generic operation `DeleteInstance`.

3128 The input parameters for this generic operation are formed as follows:

- 3129 • the `InstancePath` parameter is formed from CIM namespace, class name and key bindings,
3130 which are all formed from information units extracted from the target URI of the HTTP request
3131 (see B.1.7)

3132 This generic operation has no output parameters.

3133 Restrictions: None..

3134 **B.3.5 GET instance resource**

3135 This CIM-RS operation retrieves an instance resource (see 7.6.3), possibly including associated or
3136 referenced instance resources.

3137 If neither the `$refer` nor the `$expand` query parameter is specified, this CIM-RS operation directly maps
3138 to the generic operation `GetInstance`.

3139 The input parameters for this generic operation are formed as follows:

- 3140 • the `InstancePath` parameter is formed from CIM namespace, class name and key bindings,
3141 which are all formed from information units extracted from the target URI of the HTTP request
3142 (see B.1.2)
- 3143 • the `IncludeClassOrigin` parameter is set to false
- 3144 • the `IncludedProperties` parameter is obtained from the `$properties` query parameter as
3145 described in Table B-1

3146 The output parameters of this generic operation are used as follows:

- 3147 • the `Instance` parameter is used to form the `Instance` payload element in the HTTP
3148 response (see 7.6.1)

3149 If the `$refer` or `$expand` query parameters are specified, this CIM-RS operation maps to the generic
3150 operation `GetInstance` as described above, and possibly additional association traversal operations, as
3151 described in B.2.1.

3152 Restrictions:

- 3153 • Including the class origin of properties in the returned instance representation is not supported
3154 in CIM-RS.

3155 **B.3.6 GET page of instance collection resource**

3156 This CIM-RS operation retrieves the next page of a paged instance collection resource (see 7.8.2),
3157 resulting from enumeration by class, or from association traversal.

3158 This CIM-RS operation directly maps to the generic operation `PullInstancesWithPath`.

3159 The input parameters for this generic operation are formed as follows:

- 3160 • the `NamespacePath` parameter is formed from the CIM namespace, which is formed from
3161 information units extracted from the target URI of the HTTP request (see B.1.3 and B.1.4)
- 3162 • the `EnumerationContext` parameter is formed from the information about the next page to
3163 be retrieved within the overall collection, which is formed from information units extracted from
3164 the target URI of the HTTP request (see B.1.3 and B.1.4)

- 3165 • the `MaxObjectCount` parameter is obtained from the `$max` query parameter as described in
3166 Table B-1

3167 The output parameters of this generic operation are used as follows:

- 3168 • the `InstanceList` parameter is used to form the `instances` attribute in the
3169 `InstanceCollection` payload element in the HTTP response (see 7.8.1)
- 3170 • if the `EndOfSequence` parameter is `FALSE`, the `EnumerationContext` parameter is used to
3171 form the information about the next page to be retrieved within the overall collection, in the URI
3172 for the `next` attribute in the `InstanceCollection` payload element in the HTTP response
3173 (see 7.8.1)
- 3174 • if the `EndOfSequence` parameter is `TRUE`, the `next` attribute is omitted from the
3175 `InstanceCollection` payload element in the HTTP response (see 7.8.1)

3176 Restrictions: None.

3177 **B.3.7 GET page of reference collection resource**

3178 This CIM-RS operation retrieves the next page of a paged reference collection resource (see 7.7.2),
3179 resulting from enumeration by class, or from association traversal.

3180 This CIM-RS operation directly maps to the generic operation `PullInstancePaths`.

3181 The input parameters for this generic operation are formed as follows:

- 3182 • the `NamespacePath` parameter is formed from the CIM namespace, which is formed from
3183 information units extracted from the target URI of the HTTP request (see B.1.3 and B.1.4)
- 3184 • the `EnumerationContext` parameter is formed from the information about the next page to
3185 be retrieved within the overall collection, which is formed from information units extracted from
3186 the target URI of the HTTP request (see B.1.3 and B.1.4)
- 3187 • the `MaxObjectCount` parameter is obtained from the `$max` query parameter as described in
3188 Table B-1

3189 The output parameters of this generic operation are used as follows:

- 3190 • the `InstancePathList` parameter is used to form the `references` attribute in the
3191 `ReferenceCollection` payload element in the HTTP response (see 7.7.1)
- 3192 • if the `EndOfSequence` parameter is `FALSE`, the `EnumerationContext` parameter is used to
3193 form the information about the next page to be retrieved within the overall collection, in the URI
3194 for the `next` attribute in the `ReferenceCollection` payload element in the HTTP response
3195 (see 7.7.1)
- 3196 • if the `EndOfSequence` parameter is `TRUE`, the `next` attribute is omitted from the
3197 `ReferenceCollection` payload element in the HTTP response (see 7.7.1)

3198 Restrictions: None.

3199 **B.3.8 GET instance enumeration resource**

3200 This CIM-RS operation enumerates all instances of the specified class (including instances of subclasses)
3201 in the namespace of the targeted instance enumeration (see 7.9.1).

3202 If neither the `$refer` nor the `$expand` query parameter is specified, this CIM-RS operation directly maps
3203 to the generic operation `OpenClassInstancesWithPath`.

3204 The input parameters for this generic operation are formed as follows:

- 3205 • the `EnumClassPath` parameter is formed from:
 - 3206 – the CIM namespace, formed from information units extracted from the target URI of the
 - 3207 HTTP request (see B.1.5)
 - 3208 – the class name, obtained from the `$class` query parameter in the target URI of the HTTP
 - 3209 request (see B.1.5)
- 3210 • the `FilterQueryString` parameter is set from the `$filter` query parameter as described in
- 3211 Table B-1
- 3212 • the `FilterQueryLanguage` parameter is set to "DMTF:FQL" (see C.2)
- 3213 • the `IncludeClassOrigin` parameter is set to false
- 3214 • the `IncludedProperties` parameter is set from the `$properties` query parameter as
- 3215 described in Table B-1
- 3216 • the `ExcludeSubclassProperties` parameter is set to false
- 3217 • the `OperationTimeout` parameter is set from the `$pagingtimeout` query parameter as
- 3218 described in Table B-1
- 3219 • the `ContinueOnError` parameter is set from the `$continueonerror` query parameter as
- 3220 described in Table B-1
- 3221 • the `MaxObjectCount` parameter is set from the `$max` query parameter as described in Table
- 3222 B-1

3223 The output parameters of this generic operation are used as follows:

- 3224 • the `InstanceList` parameter is used to form the `instances` attribute in the
- 3225 `InstanceCollection` payload element in the HTTP response (see 7.8.1)
- 3226 • if the `EndOfSequence` parameter is FALSE, the `EnumerationContext` parameter is used to
- 3227 form the information about the next page to be retrieved within the overall collection, in the URI
- 3228 for the `next` attribute in the `InstanceCollection` payload element in the HTTP response
- 3229 (see 7.8.1)
- 3230 • if the `EndOfSequence` parameter is TRUE, the `next` attribute is omitted from the
- 3231 `InstanceCollection` payload element in the HTTP response (see 7.8.1)

3232 If the `$refer` or `$expand` query parameters are specified, this CIM-RS operation maps to the generic
 3233 operation `OpenClassInstancesWithPath` as described above, and possibly additional association
 3234 traversal operations, as described in B.2.1.

3235 Restrictions:

- 3236 • Including the class origin of properties in the returned instance representations is not supported
- 3237 in CIM-RS.
- 3238 • Excluding subclass properties in the returned instance representations by setting a single
- 3239 indicator is not supported in CIM-RS (they can be excluded through the `$properties` query
- 3240 parameter).

3241 **B.3.9 GET server entry point resource**

3242 This CIM-RS operation retrieves the server entry point resource (see 7.12.2), which describes optional
 3243 capabilities of the CIM-RS support, and information about the CIM namespaces of the server.

- 3244 This CIM-RS operation does not map to any generic operation.
- 3245 The CIM namespaces can be determined through the generic operation `GetInstance` on class
3246 `CIM_Namespace` in the Interop namespace. Alternatively, this information can be retrieved through direct
3247 interfaces.
- 3248 Restrictions: None.
- 3249 **B.3.10 PUT instance resource**
- 3250 This CIM-RS operation modifies some or all property values of an instance resource (see 7.6.4).
- 3251 This CIM-RS operation directly maps to the generic operation `ModifyInstance`.
- 3252 The input parameters for this generic operation are formed as follows:
- 3253 • the `InstancePath` parameter is formed from CIM namespace, class name and key bindings,
3254 which are all formed from information units extracted from the target URI of the HTTP request
3255 (see B.1.2)
 - 3256 • the `ModifiedInstance` parameter is formed from the `instance` attribute of the `Instance`
3257 payload element in the HTTP request (see 7.6.1)
 - 3258 • the `IncludedProperties` parameter is obtained from the `$properties` query parameter as
3259 described in Table B-1
- 3260 This generic operation does not have any output parameters.
- 3261 Restrictions: None.
- 3262 **B.4 Listener operations**
- 3263 This subclause describes a listener's decision tree for how incoming CIM-RS listener operations are to be
3264 analyzed, identified, and mapped to generic listener operations: For each HTTP method, the listener will
3265 examine its target URI. Based upon the listener's defined URI structure, it will determine what type of
3266 resource is targeted, and will then determine which generic operations are to be invoked.
- 3267 The following subclauses describe each combination of HTTP method and resource type.
- 3268 **B.4.1 POST listener destination resource**
- 3269 This CIM-RS listener operation delivers an indication to a listener (see 7.11.2).
- 3270 This CIM-RS operation directly maps to the generic operation `DeliverIndication`.
- 3271 The input parameters for this generic operation are formed as follows:
- 3272 • the `ListenerDestination` parameter is formed from information units extracted from the
3273 target URI of the HTTP request (see B.1.8)
 - 3274 • the `Indication` parameter is formed from the `indication` attribute of the
3275 `IndicationDeliveryRequest` payload element in the HTTP request (see 7.11.1)
- 3276 This generic operation does not have any output parameters.
- 3277 Restrictions: None.

3278 B.4.2 GET listener entry point resource

3279 This CIM-RS operation retrieves the listener entry point resource (see 7.13.2), which describes optional
3280 capabilities of the CIM-RS support.

3281 This CIM-RS operation does not map to any generic operation.

3282 Restrictions: None.

ANNEX C (informative)

3283
3284
3285
3286

Mapping generic operations to CIM-RS

3287 This annex describes how generic operations (see [DSP0223](#)) are to be mapped to CIM-RS operations,
3288 resources, and query parameters. This mapping is provided primarily to describe how the CIM-RS
3289 protocol conforms to generic operations. This mapping can also be used to translate operation
3290 requirements defined in management profiles that are stated in terms of generic operations, into CIM-RS
3291 operations. The latter may be useful for implementations of CIM servers that define their provider API in
3292 terms of CIM-RS operations.

3293 **C.1 Conformance**

3294 CIM-RS does not satisfy all conformance requirements defined in generic operations ([DSP0223](#)). As a
3295 result, CIM-RS is not a conforming WBEM protocol. The subclauses in this annex provide details.

3296 **C.2 Support of optional generic operations features**

3297 This subclause describes how CIM-RS supports optional features defined in generic operations.

- 3298 • CIM-RS does not support client side control of returning class origin information (generic
3299 operation parameter `IncludeClassOrigin`)
- 3300 • CIM-RS supports error handling by means of returning DMTF standard messages (also known
3301 as "extended error handling")
- 3302 • CIM-RS supports filter queries in pulled instance enumeration operations. However, only the
3303 upcoming DMTF *Filter Query Language* will be supported. In anticipation of that, the
3304 `FilterQueryLanguage` parameter of any generic operations is set to "`DMTF:FQL`".
- 3305 • CIM-RS supports client side control of continuation on error for pulled instance enumeration
3306 operations

3307 **C.3 Operations supported**

3308 This subclause describes generic operations that are supported in CIM-RS.

3309 **C.3.1 GetInstance**

3310 This generic operation is supported via HTTP GET on an instance resource (see 7.6.3).

3311 Its input parameters map to CIM-RS as follows:

- 3312 • `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3313 • `IncludeClassOrigin`: Not supported in CIM-RS (optional in [DSP0223](#))
- 3314 • `IncludedProperties`: `$properties` query parameter (see Table B-1)

3315 Its output parameters map to CIM-RS as follows:

- 3316 • `Instance`: `Instance` payload element in HTTP response (see 7.6.1)

3317 Conformance: Yes.

3318 **C.3.2 DeleteInstance**

3319 This generic operation is supported via HTTP DELETE on an instance resource (see 0).

3320 Its input parameters map to CIM-RS as follows:

- 3321 • `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)

3322 This generic operation has no output parameters.

3323 Conformance: Yes.

3324 **C.3.3 ModifyInstance**

3325 This generic operation is supported via HTTP PUT on an instance resource (see 7.6.4).

3326 Its input parameters map to CIM-RS as follows:

- 3327 • `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3328 • `ModifiedInstance`: Instance payload element in HTTP request (see 7.6.1)
- 3329 • `IncludedProperties`: `$properties` query parameter (see Table B-1)

3330 This generic operation has no output parameters.

3331 Conformance: Yes.

3332 **C.3.4 CreateInstance**

3333 This generic operation is supported via HTTP POST on an instance creation resource (see 7.5.1).

3334 Its input parameters map to CIM-RS as follows:

- 3335 • `ClassPath`: Information units in target URI of the HTTP request (see B.1.1)
- 3336 • `NewInstance`: Instance payload element in HTTP request (see 7.6.1)

3337 Its output parameters map to CIM-RS as follows:

- 3338 • `InstancePath`: Location header field in HTTP response (see 7.5.1)

3339 Conformance: Yes.

3340 **C.3.5 OpenClassInstancesWithPath**

3341 This generic operation is supported via HTTP GET on an instance enumeration resource (see 7.9.1).

3342 Its input parameters map to CIM-RS as follows:

- 3343 • `EnumClassPath`: Information units in target URI of the HTTP request (see B.1.5)
- 3344 • `FilterQueryString`: `$filter` query parameter (see Table B-1)
- 3345 • `FilterQueryLanguage`: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- 3346 • `IncludeClassOrigin`: Not supported in CIM-RS (optional in [DSP0223](#))
- 3347 • `IncludedProperties`: `$properties` query parameter (see Table B-1)
- 3348 • `ExcludeSubclassProperties`: Not supported directly; can be achieved with `$properties`
- 3349 • query parameter (see Table B-1)

- 3350 • `OperationTimeout: $pagingtimeout` query parameter (see Table B-1)
- 3351 • `ContinueOnError: $continueonerror` query parameter (see Table B-1)
- 3352 • `MaxObjectCount: $max` query parameter (see Table B-1)

3353 Its output parameters map to CIM-RS as follows:

- 3354 • `InstanceList: instances` attribute of `InstanceCollection` payload element in HTTP
3355 response (see 7.8.1)
- 3356 • `EnumerationContext: information` units in URI of `next` attribute of `InstanceCollection`
3357 payload element in HTTP response (see 7.8.1)
- 3358 • `EndOfSequence: omission` or presence of `next` attribute of `InstanceCollection` payload
3359 element in HTTP response (see 7.8.1)

3360 Conformance: Yes.

3361 C.3.6 OpenClassInstancePaths

3362 This generic operation is supported via HTTP GET on an instance enumeration resource (see 7.9.1),
3363 where its `$properties` query parameter is set to include no properties.

3364 Its input parameters map to CIM-RS as follows:

- 3365 • `EnumClassPath: Information` units in target URI of the HTTP request (see B.1.5)
- 3366 • `FilterQueryString: $filter` query parameter (see Table B-1)
- 3367 • `FilterQueryLanguage: Only "DMTF:FQL"` is supported by CIM-RS (see C.2)
- 3368 • `OperationTimeout: $pagingtimeout` query parameter (see Table B-1)
- 3369 • `ContinueOnError: $continueonerror` query parameter (see Table B-1)
- 3370 • `MaxObjectCount: $max` query parameter (see Table B-1)

3371 Its output parameters map to CIM-RS as follows:

- 3372 • `InstancePathList: instances` attribute of `InstanceCollection` payload element in
3373 HTTP response (see 7.8.1)
- 3374 • `EnumerationContext: information` units in URI of `next` attribute of `InstanceCollection`
3375 payload element in HTTP response (see 7.8.1)
- 3376 • `EndOfSequence: omission` or presence of `next` attribute of `InstanceCollection` payload
3377 element in HTTP response (see 7.8.1)

3378 Conformance: Yes.

3379 C.3.7 OpenAssociatedInstancesWithPath

3380 This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a
3381 `$properties` query parameter that specifies not to include any properties, and with a `$expand` query
3382 parameter that specifies each association to be traversed (for example,
3383 `$expand=AssociationClassName.[AssociatedClassName]AssociatedRoleName`).

3384 Its input parameters map to CIM-RS as follows:

- 3385 • `SourceInstancePath: Information` units in target URI of the HTTP request (see B.1.2)

- 3386 • AssociationClassName: association class in \$expand query parameter (see B.2.1)
- 3387 • AssociatedClassName: associated class filter in \$expand query parameter (see B.2.1)
- 3388 • SourceRoleName: Not supported in CIM-RS (mandatory in [DSP0223](#))
- 3389 • AssociatedRoleName: association end in \$expand query parameter (see Table B-1)
- 3390 • FilterQueryString: \$filter query parameter (see Table B-1)
- 3391 • FilterQueryLanguage: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- 3392 • IncludeClassOrigin: Not supported in CIM-RS (optional in [DSP0223](#))
- 3393 • IncludedProperties: \$properties query parameter (see Table B-1) specifying properties
- 3394 in the navigation properties included via the \$expand query parameter
- 3395 • ExcludeSubclassProperties: Not supported directly; can be achieved with the
- 3396 \$properties query parameter (see Table B-1) specifying properties in the navigation
- 3397 properties included via the \$expand query parameter
- 3398 • OperationTimeout: \$pagingtimeout query parameter (see Table B-1)
- 3399 • ContinueOnError: \$continueonerror query parameter (see Table B-1)
- 3400 • MaxObjectCount: \$max query parameter (see Table B-1)

3401 Its output parameters map to CIM-RS as follows:

- 3402 • InstanceList: instances attribute of InstanceCollection payload element in HTTP
- 3403 response (see 7.8.1)
- 3404 • EnumerationContext: information units in URI of next attribute of InstanceCollection
- 3405 payload element in HTTP response (see 7.8.1)
- 3406 • EndOfSequence: omission or presence of next attribute of InstanceCollection payload
- 3407 element in HTTP response (see 7.8.1)

3408 Conformance: No, for the following reasons:

- 3409 • the mandatory SourceRoleName filter is not supported
- 3410 • traversal of all referencing associations without knowing them upfront is not supported

3411 C.3.8 OpenAssociatedInstancePaths

3412 This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a
 3413 \$properties query parameter that specifies not to include any properties, and with a \$refer query
 3414 parameter that specifies each association to be traversed (for example,
 3415 \$refer=AssociationClassName.[AssociatedClassName]AssociatedRoleName).

3416 Its input parameters map to CIM-RS as follows:

- 3417 • SourceInstancePath: Information units in target URI of the HTTP request (see B.1.2)
- 3418 • AssociationClassName: association class in \$refer query parameter (see B.2.1)
- 3419 • AssociatedClassName: associated class filter in \$refer query parameter (see B.2.1)
- 3420 • SourceRoleName: Not supported in CIM-RS (mandatory in [DSP0223](#))
- 3421 • AssociatedRoleName: association end in \$refer query parameter (see B.2.1)

- 3422 • `FilterQueryString`: `$filter` query parameter (see Table B-1)
- 3423 • `FilterQueryLanguage`: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- 3424 • `IncludeClassOrigin`: Not supported in CIM-RS (optional in [DSP0223](#))
- 3425 • `IncludedProperties`: `$properties` query parameter (see Table B-1) specifying properties
3426 in the navigation properties included via the `$refer` query parameter
- 3427 • `ExcludeSubclassProperties`: Not supported directly; can be achieved with the
3428 `$properties` query parameter (see Table B-1) specifying properties in the navigation
3429 properties included via the `$refer` query parameter
- 3430 • `OperationTimeout`: `$pagingtimeout` query parameter (see Table B-1)
- 3431 • `ContinueOnError`: `$continueonerror` query parameter (see Table B-1)
- 3432 • `MaxObjectCount`: `$max` query parameter (see Table B-1)

3433 Its output parameters map to CIM-RS as follows:

- 3434 • `InstancePathList`: `instances` attribute of `InstanceCollection` payload element in
3435 HTTP response (see 7.8.1)
- 3436 • `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection`
3437 payload element in HTTP response (see 7.8.1)
- 3438 • `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload
3439 element in HTTP response (see 7.8.1)

3440 Conformance: No, for the following reasons:

- 3441 • the mandatory `SourceRoleName` filter is not supported
- 3442 • traversal of all referencing associations without knowing them upfront is not supported

3443 C.3.9 OpenReferencingInstancesWithPath

3444 This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a
3445 `$properties` query parameter that specifies not to include any properties, and with a `$expand` query
3446 parameter that specifies each association to be returned (for example,
3447 `$expand=AssociationClassName`).

3448 Its input parameters map to CIM-RS as follows:

- 3449 • `SourceInstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3450 • `AssociationClassName`: association class in `$expand` query parameter (see B.2.1)
- 3451 • `AssociatedClassName`: associated class filter in `$expand` query parameter (see B.2.1)
- 3452 • `SourceRoleName`: Not supported in CIM-RS (mandatory in [DSP0223](#))
- 3453 • `AssociatedRoleName`: association end in `$expand` query parameter (see B.2.1)
- 3454 • `FilterQueryString`: `$filter` query parameter (see Table B-1)
- 3455 • `FilterQueryLanguage`: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- 3456 • `IncludeClassOrigin`: Not supported in CIM-RS (optional in [DSP0223](#))
- 3457 • `IncludedProperties`: `$properties` query parameter (see Table B-1) specifying properties
3458 in the navigation properties included via the `$expand` query parameter

- 3459 • ExcludeSubclassProperties: Not supported directly; can be achieved with the
- 3460 \$properties query parameter (see Table B-1) specifying properties in the navigation
- 3461 properties included via the \$expand query parameter
- 3462 • OperationTimeout: \$pagingtimeout query parameter (see Table B-1)
- 3463 • ContinueOnError: \$continueonerror query parameter (see Table B-1)
- 3464 • MaxObjectCount: \$max query parameter (see Table B-1)

3465 Its output parameters map to CIM-RS as follows:

- 3466 • InstanceList: instances attribute of InstanceCollection payload element in HTTP
- 3467 response (see 7.8.1)
- 3468 • EnumerationContext: information units in URI of next attribute of InstanceCollection
- 3469 payload element in HTTP response (see 7.8.1)
- 3470 • EndOfSequence: omission or presence of next attribute of InstanceCollection payload
- 3471 element in HTTP response (see 7.8.1)

3472 Conformance: No, for the following reasons:

- 3473 • the mandatory SourceRoleName filter is not supported
- 3474 • return of all referencing associations without knowing them upfront is not supported

3475 C.3.10 OpenReferencingInstancePaths

3476 This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a

3477 \$properties query parameter that specifies not to include any properties, and with a \$refer query

3478 parameter that specifies each association to be returned (for example,

3479 \$refer=AssociationClassName).

3480 Its input parameters map to CIM-RS as follows:

- 3481 • SourceInstancePath: Information units in target URI of the HTTP request (see B.1.2)
- 3482 • AssociationClassName: association class in \$refer query parameter (see B.2.1)
- 3483 • AssociatedClassName: associated class filter in \$refer query parameter (see B.2.1)
- 3484 • SourceRoleName: Not supported in CIM-RS (mandatory in [DSP0223](#))
- 3485 • AssociatedRoleName: association end in \$refer query parameter (see B.2.1)
- 3486 • FilterQueryString: \$filter query parameter (see Table B-1)
- 3487 • FilterQueryLanguage: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- 3488 • IncludeClassOrigin: Not supported in CIM-RS (optional in [DSP0223](#))
- 3489 • IncludedProperties: \$properties query parameter (see Table B-1) specifying properties
- 3490 in the navigation properties included via the \$refer query parameter
- 3491 • ExcludeSubclassProperties: Not supported directly; can be achieved with the
- 3492 \$properties query parameter (see Table B-1) specifying properties in the navigation
- 3493 properties included via the \$refer query parameter
- 3494 • OperationTimeout: \$pagingtimeout query parameter (see Table B-1)
- 3495 • ContinueOnError: \$continueonerror query parameter (see Table B-1)

- 3496
- `MaxObjectCount`: `$max` query parameter (see Table B-1)

3497 Its output parameters map to CIM-RS as follows:

- 3498
- `InstancePathList`: `instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3500
- `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3502
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3503

3504 Conformance: No, for the following reasons:

- 3505
- the mandatory `SourceRoleName` filter is not supported
- 3506
- return of all referencing associations without knowing them upfront is not supported

3507 C.3.11 PullInstancesWithPath

3508 This generic operation is supported via HTTP GET on a page of an instance collection resource (see
3509 7.8.2), that had been created (via the `$properties` query parameter) such that properties were to be
3510 returned.

3511 Its input parameters map to CIM-RS as follows:

- 3512
- `NamespacePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3513
- `EnumerationContext`: information units in target URI of the HTTP request (see B.1.2)
- 3514
- `MaxObjectCount`: `$max` query parameter (see Table B-1)

3515 Its output parameters map to CIM-RS as follows:

- 3516
- `InstanceList`: `instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3517
- `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3518
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3519
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3520
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3521

3522 Conformance: Yes.

3523 C.3.12 PullInstancePaths

3524 This generic operation is supported via HTTP GET on a page of an instance collection resource (see
3525 7.8.2), that had been created (via the `$properties` query parameter) such that no properties were to be
3526 returned.

3527 Its input parameters map to CIM-RS as follows:

- 3528
- `NamespacePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3529
- `EnumerationContext`: information units in target URI of the HTTP request (see B.1.2)
- 3530
- `MaxObjectCount`: `$max` query parameter (see Table B-1)

3531 Its output parameters map to CIM-RS as follows:

- 3532 • InstanceList: instances attribute of InstanceCollection payload element in HTTP
3533 response (see 7.8.1)
- 3534 • EnumerationContext: information units in URI of next attribute of InstanceCollection
3535 payload element in HTTP response (see 7.8.1)
- 3536 • EndOfSequence: omission or presence of next attribute of InstanceCollection payload
3537 element in HTTP response (see 7.8.1)

3538 Conformance: Yes.

3539 **C.3.13 InvokeMethod**

3540 This generic operation is supported via HTTP POST on a non-static method invocation resource (see
3541 7.10.3).

3542 Its input parameters map to CIM-RS as follows:

- 3543 • InstancePath: Information units in target URI of the HTTP request (see B.1.2)
- 3544 • MethodName: method attribute of MethodRequest payload element in HTTP request (see
3545 7.10.1)
- 3546 • InParmValues: parameters attribute of MethodRequest payload element in HTTP request
3547 (see 7.10.1)

3548 Its output parameters map to CIM-RS as follows:

- 3549 • OutParmValues: parameters attribute of MethodResponse payload element in HTTP
3550 response (see 7.10.2)
- 3551 • ReturnValue: returnvalue attribute of MethodResponse payload element in HTTP
3552 response (see 7.10.2)

3553 Conformance: Yes.

3554 **C.3.14 InvokeStaticMethod**

3555 This generic operation is supported via HTTP POST on a static method invocation resource (see 7.10.3).

3556 Its input parameters map to CIM-RS as follows:

- 3557 • ClassPath: Information units in target URI of the HTTP request (see B.1.2)
- 3558 • MethodName: method attribute of MethodRequest payload element in HTTP request (see
3559 7.10.1)
- 3560 • InParmValues: parameters attribute of MethodRequest payload element in HTTP request
3561 (see 7.10.1)

3562 Its output parameters map to CIM-RS as follows:

- 3563 • OutParmValues: parameters attribute of MethodResponse payload element in HTTP
3564 response (see 7.10.2)
- 3565 • ReturnValue: returnvalue attribute of MethodResponse payload element in HTTP
3566 response (see 7.10.2)

3567 Conformance: Yes.

3568 **C.4 Operations not supported**

3569 The following generic operations are not supported in CIM-RS.

3570 **C.4.1 Direct instance enumeration operations**

3571 Direct instance enumeration operations are not supported in CIM-RS, because it is always possible that
 3572 the resulting collections in CIM-RS are paged.

3573 **Table C-1 – Pulled equivalents of direct instance enumeration operations**

Unsupported Direct Enumeration Operation	Supported Pulled Equivalent
GetClassInstancesWithPath	OpenClassInstancesWithPath (Section C.3.5)
GetClassInstancePaths	OpenClassInstancePaths (Section C.3.6)
GetAssociatedInstancesWithPath	OpenAssociatedInstancesWithPath (Section C.3.7)
GetAssociatedInstancePaths	OpenAssociatedInstancePaths (Section C.3.8)
GetReferencingInstancesWithPath	OpenReferencingInstancesWithPath (Section C.3.9)
GetReferencingInstancePaths	OpenReferencingInstancePaths (Section C.3.10)

3574

3575 **C.4.2 Class and qualifier type operations**

3576 Class and qualifier type operations are not supported in CIM-RS.

- 3577 • GetClass
- 3578 • DeleteClass
- 3579 • ModifyClass
- 3580 • CreateClass
- 3581 • GetTopClassesWithPath
- 3582 • GetTopClassPaths
- 3583 • GetSubClassesWithPath
- 3584 • GetSubClassPaths
- 3585 • GetAssociatedClassesWithPath
- 3586 • GetAssociatedClassPaths
- 3587 • GetReferencingClassesWithPath
- 3588 • GetReferencingInstancePaths
- 3589 • GetQualifierType
- 3590 • DeleteQualifierType
- 3591 • CreateQualifierType
- 3592 • EnumerateQualifierTypesWithPath

3593 C.4.3 Other operations

3594 The following other generic operations are not supported in CIM-RS.

- 3595 • OpenQueryInstances
- 3596 • PullInstances
- 3597 • EnumerationCount
- 3598 • CloseEnumeration

ANNEX D (informative)

Examples

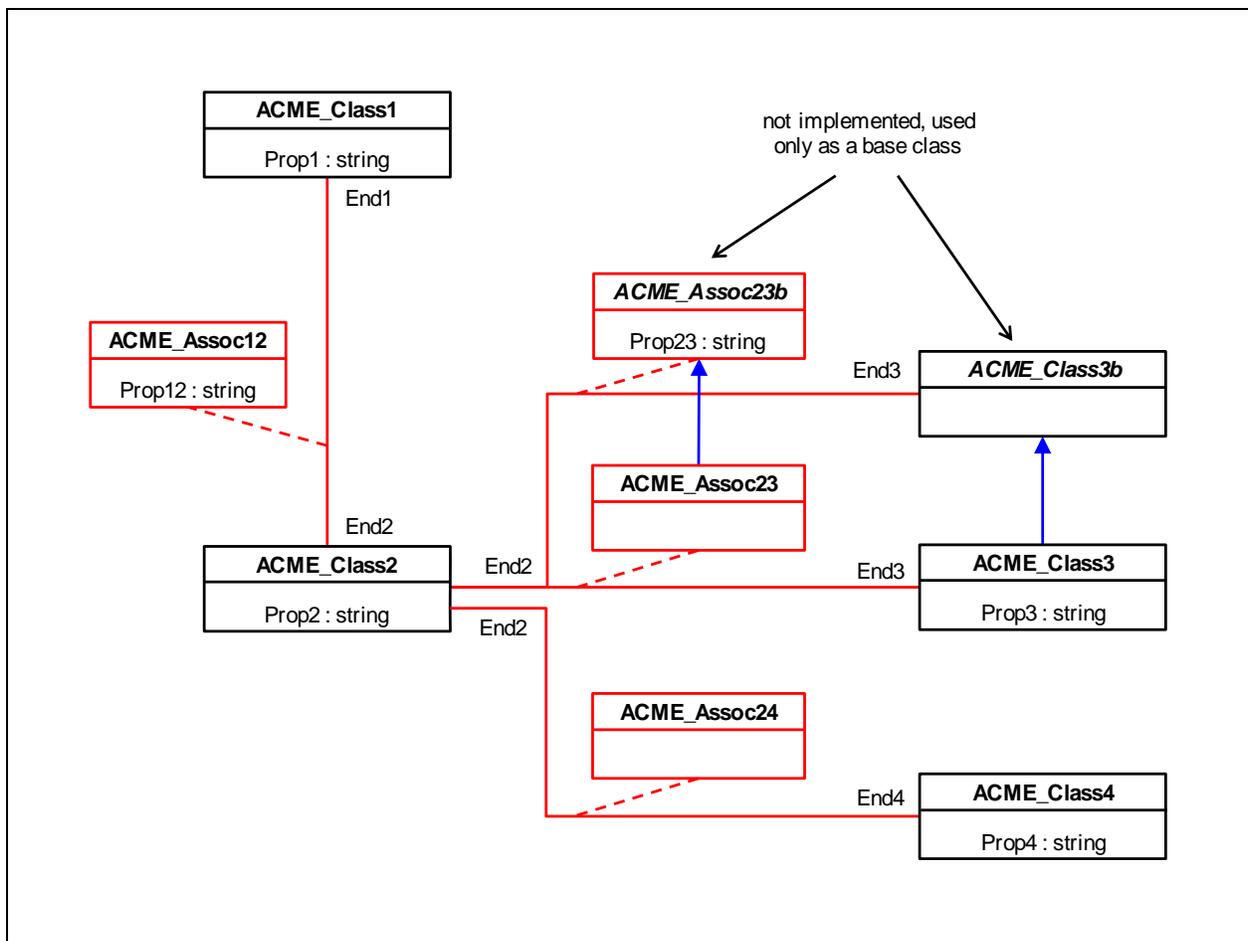
3599
3600
3601
3602

D.1 Navigation between resources

3604 This annex provides examples on how to navigate between resources using the \$expand (see 6.5.3) and
3605 \$refer (see 6.5.9) query parameters. For a description of the concepts for navigating between resources,
3606 see 5.6.

D.1.1 Classes and instances used in the examples

3608 The examples use the classes from the class diagram shown in Figure D-1.



3609
3610

Figure D-1 – Class diagram for navigation examples

3612 The representations of results uses an informal notation that indicates nesting of elements by indentation.

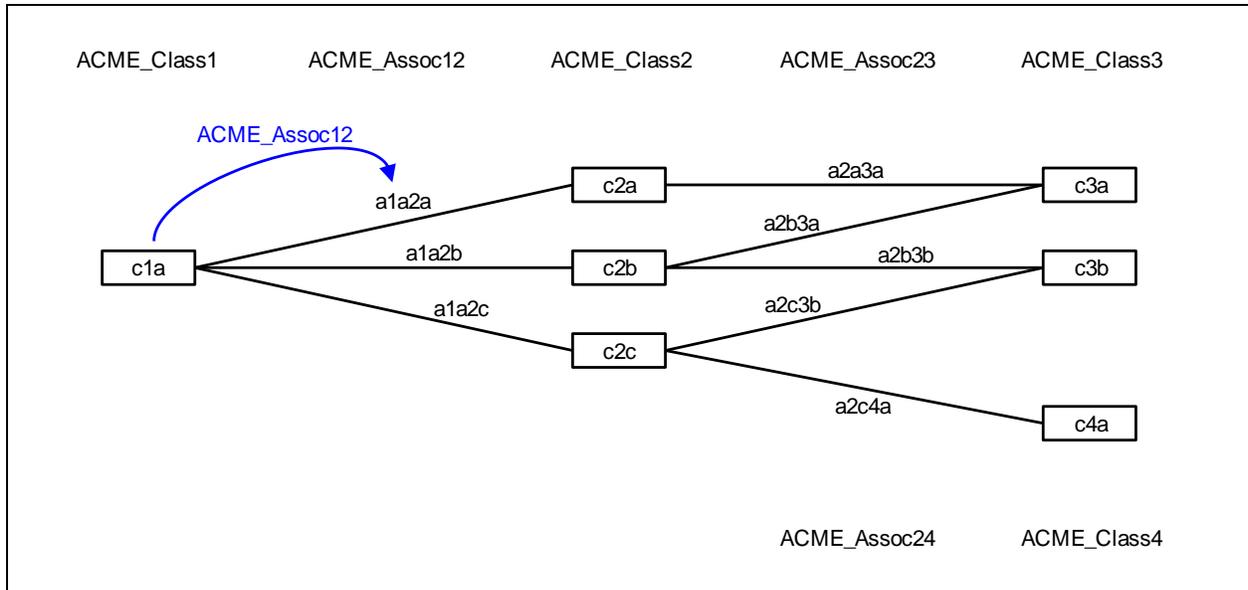
3613 The examples are limited to requests for instance retrieval, for brevity. Requests for retrieval of instance
3614 collections work the same way, except that each instance in the collection is affected.

3615 The following MOF defines the classes shown in Figure D-1:

```
3616 class ACME_Class1 { string Prop1; };
3617
3618 class ACME_Class2 { string Prop2; };
3619
3620 [Abstract]
3621 class ACME_Class3b { }; // not implemented
3622
3623 class ACME_Class3 : ACME_Class3b { string Prop3; };
3624
3625 [Association]
3626 class ACME_Assoc12 {
3627     ACME_Class1 REF End1;
3628     ACME_Class2 REF End2;
3629     string Prop12;
3630 };
3631
3632 [Association, Abstract]
3633 class ACME_Assoc23b { // not implemented
3634     ACME_Class2 REF End2;
3635     ACME_Class3b REF End3;
3636     string Prop23;
3637 };
3638
3639 [Association]
3640 class ACME_Assoc23 : ACME_Assoc23b {
3641     [Override("End3")] ACME_Class3 REF End3; // now references the subclass
3642 };
3643
3644 [Association]
3645 class ACME_Assoc24 {
3646     ACME_Class2 REF End2;
3647     ACME_Class4 REF End4;
3648 };
```

3649 **D.1.2 Navigation to referencing association instances**

3650 In this example, the client retrieves an instance and specifies a navigation path that identifies association
 3651 instances that reference the instance being retrieved. Figure D-2 shows the instance diagram and the
 3652 blue navigation path "ACME_Assoc12", starting at instance c1a.



3653
3654

3655 **Figure D-2 – Example instance diagram for navigation to referencing association instances**

3656 An instance retrieval request using this navigation path with the \$refer query parameter will return the
 3657 following instance representation:

```

3658 GET /c1a?$refer=ACME_Assoc12
3659
3660 Instance c1a:
3661   Prop1: "...
3662   ACME_Assoc12: ReferenceCollection:
3663     ref a1a2a
3664     ref a1a2b
3665     ref a1a2c
    
```

3666 An instance retrieval request using this navigation path with the \$expand query parameter will return the
 3667 following instance representation:

```

3668 GET /c1a?$expand=ACME_Assoc12
3669
3670 Instance c1a:
3671   Prop1: "...
3672   ACME_Assoc12: InstanceCollection:
3673     Instance a1a2a:
3674       End1: ref c1a
3675       End2: ref c2a
3676       Prop12: "...
3677     Instance a1a2b:
    
```

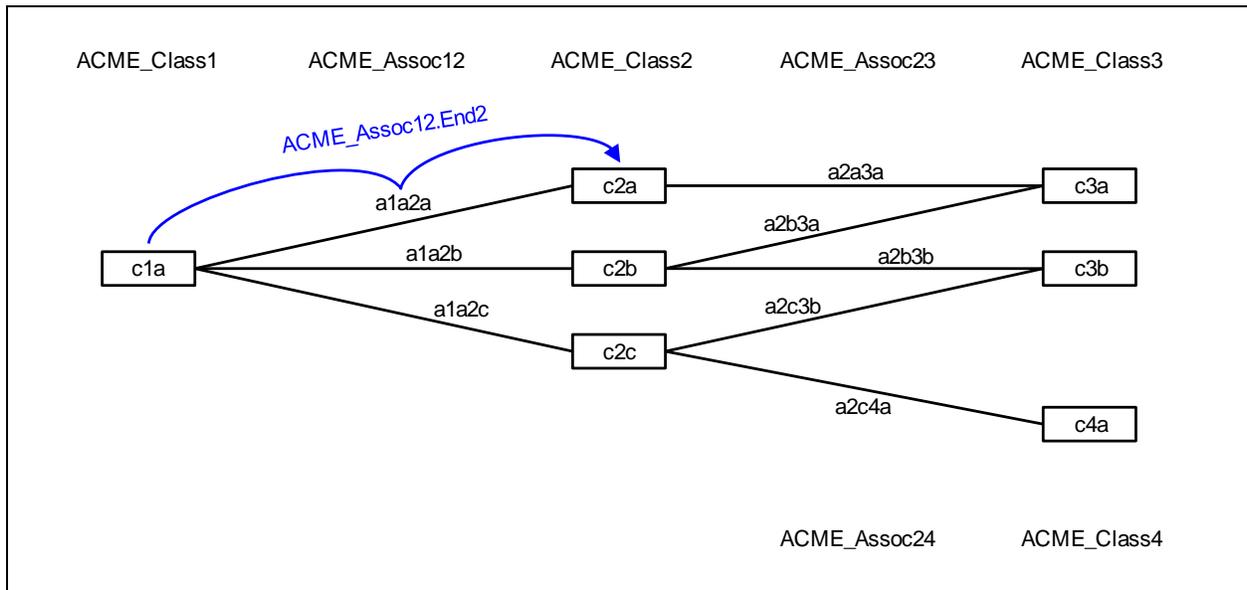
```

3678     End1: ref c1a
3679     End2: ref c2b
3680     Prop12: "..."/>

```

3685 D.1.3 Navigation to associated instances

3686 In this example, the client retrieves an instance and specifies a navigation path that identifies the
 3687 instances associated to the instance being retrieved. Figure D-3 shows the instance diagram and the blue
 3688 navigation path "ACME_Assoc12.End2", starting at instance c1a.



3689
3690

3691 **Figure D-3 – Example instance diagram for navigation to associated instances**

3692 An instance retrieval request using this navigation path with the \$refer query parameter will return the
 3693 following instance representation:

```

3694 GET /c1a?$refer=ACME_Assoc12.End2
3695
3696 Instance c1a:
3697     Prop1: "..."/>

```

3702 An instance retrieval request using this navigation path with the \$expand query parameter will return the
 3703 following instance representation:

```

3704 GET /c1a?$expand=ACME_Assoc12.End2
3705
3706 Instance c1a:

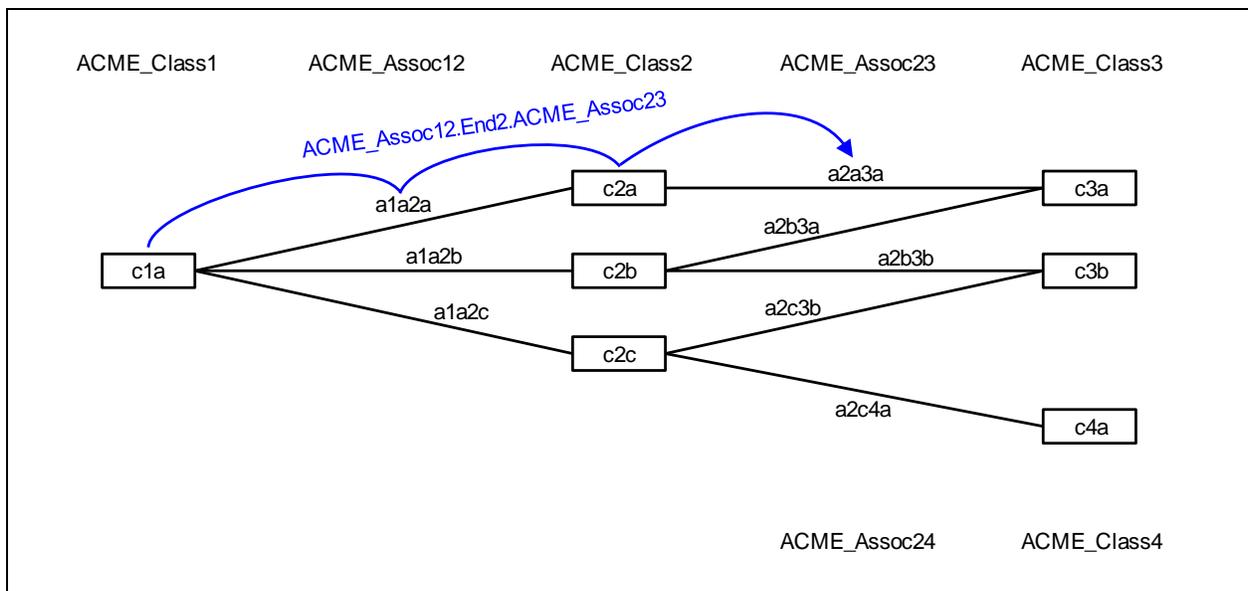
```

```

3707 Prop1: "...
3708 ACME_Assoc12.End2: InstanceCollection:
3709 Instance c2a:
3710 Prop2: "...
3711 Instance c2b:
3712 Prop2: "...
3713 Instance c2c:
3714 Prop2: "...
    
```

3715 **D.1.4 Navigation to association instances across one hop**

3716 In this example, the client retrieves an instance and specifies a navigation path that identifies the
 3717 association instances that reference the instances associated to the instance being retrieved. Figure D-4
 3718 shows the instance diagram and the blue navigation path "ACME_Assoc12.End2.ACME_Assoc23",
 3719 starting at instance c1a.



3720
3721

3722 **Figure D-4 – Example instance diagram for navigation to association instances across one hop**

3723 An instance retrieval request using this navigation path with the \$refer query parameter will return the
 3724 following instance representation:

```

3725 GET /c1a?$refer=ACME_Assoc12.End2.ACME_Assoc23
3726
3727 Instance c1a:
3728 Prop1: "...
3729 ACME_Assoc12.End2.ACME_Assoc23: ReferenceCollection:
3730 ref a2a3a
3731 ref a2b3a
3732 ref a2b3b
3733 ref a2c3b
    
```

3734 Note that instances of association class ACME_Assoc24 are not included, because navigation across
3735 ACME_Assoc23 was requested.

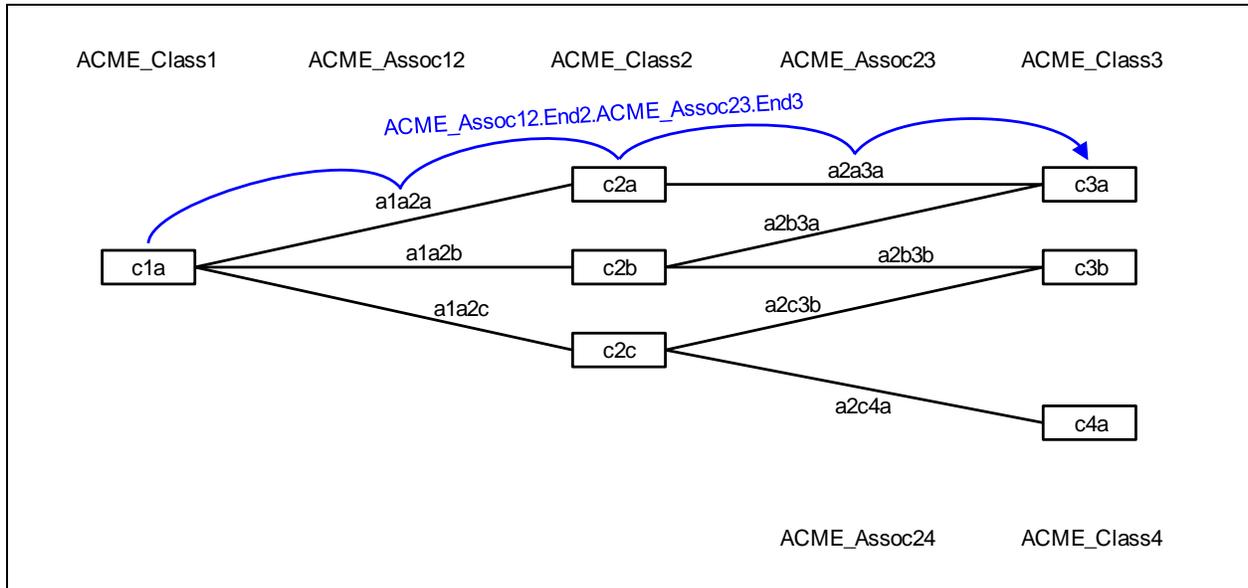
3736 An instance retrieval request using this navigation path with the \$expand query parameter will return the
3737 following instance representation:

```
3738 GET /c1a?$expand=ACME_Assoc12.End2.ACME_Assoc23
3739
3740 Instance c1a:
3741   Prop1: "..."/>

```

3759 **D.1.5 Navigation to associated instances across two hops**

3760 In this example, the client retrieves an instance and specifies a navigation path that identifies instances
 3761 associated to the instance being retrieved across two specific association hops. Figure D-5 shows the
 3762 instance diagram and the blue navigation path "ACME_Assoc12.End2.ACME_Assoc23.End3", starting at
 3763 instance c1a.



3764
3765

3766 **Figure D-5 – Example instance diagram for navigation to associated instances across two hops**

3767 An instance retrieval request using this navigation path with the \$refer query parameter will return the
 3768 following instance representation:

```
3769 GET /c1a?$refer=ACME_Assoc12.End2.ACME_Assoc23.End3
3770
3771 Instance c1a:
3772   Prop1: "...
3773   ACME_Assoc12.End2.ACME_Assoc23.End3: ReferenceCollection:
3774     ref c3a
3775     ref c3a
3776     ref c3b
3777     ref c3b
```

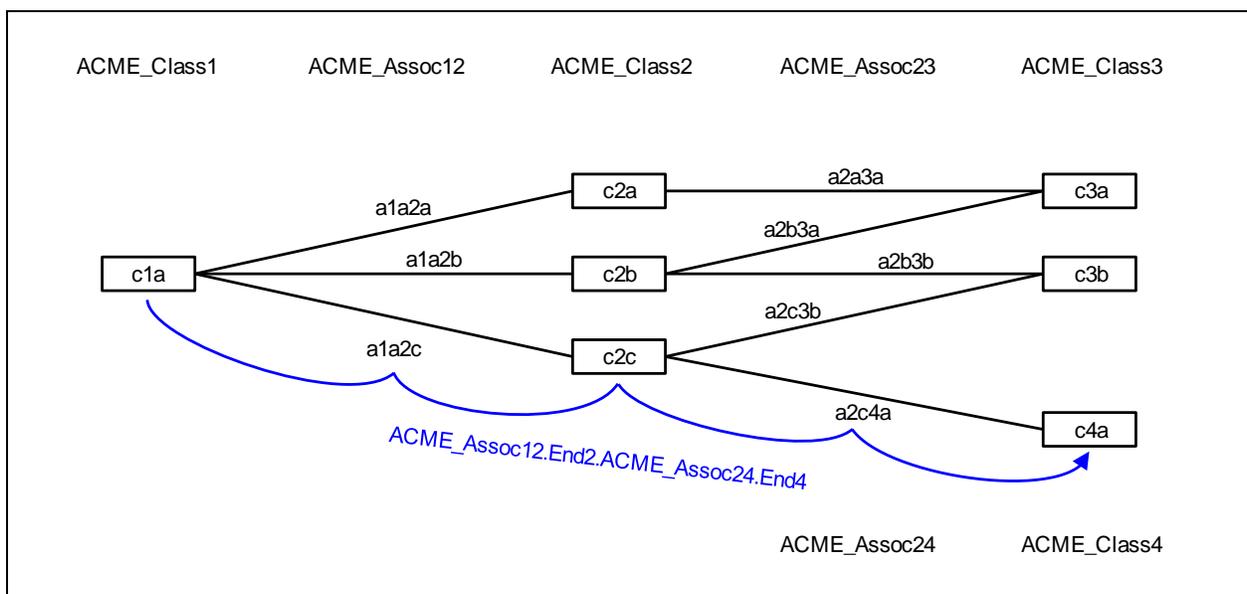
3778 Note that instances c3a and c3b each occur two times in the list. The reason for this is that the inclusion
 3779 is driven strictly by the navigation paths that lead to the desired target, and there is no optimization to
 3780 reduce any duplicates.

3781 Note that instances of class ACME_Class4 are not included, because navigation across ACME_Assoc23
 3782 and its End3 was requested.

3783 An instance retrieval request using this navigation path with the \$expand query parameter will also return
 3784 the same duplicates and is not shown, for brevity.

3785 **D.1.6 Navigation to associated instances across two hops (2)**

3786 This example is similar to the previous example, except that the navigation path uses the other possible
 3787 association for the second hop. Figure D-6 shows the instance diagram and the blue navigation path
 3788 "ACME_Assoc12.End2.ACME_Assoc24.End4", starting at instance c1a.



3789
 3790

3791 **Figure D-6 – Example instance diagram for navigation to associated instances across two hops (2)**

3792 An instance retrieval request using this navigation path with the \$refer query parameter will return the
 3793 following instance representation:

```
3794 GET /c1a?$refer=ACME_Assoc12.End2.ACME_Assoc24.End4
3795
3796 Instance c1a:
3797   Prop1: "..."/>

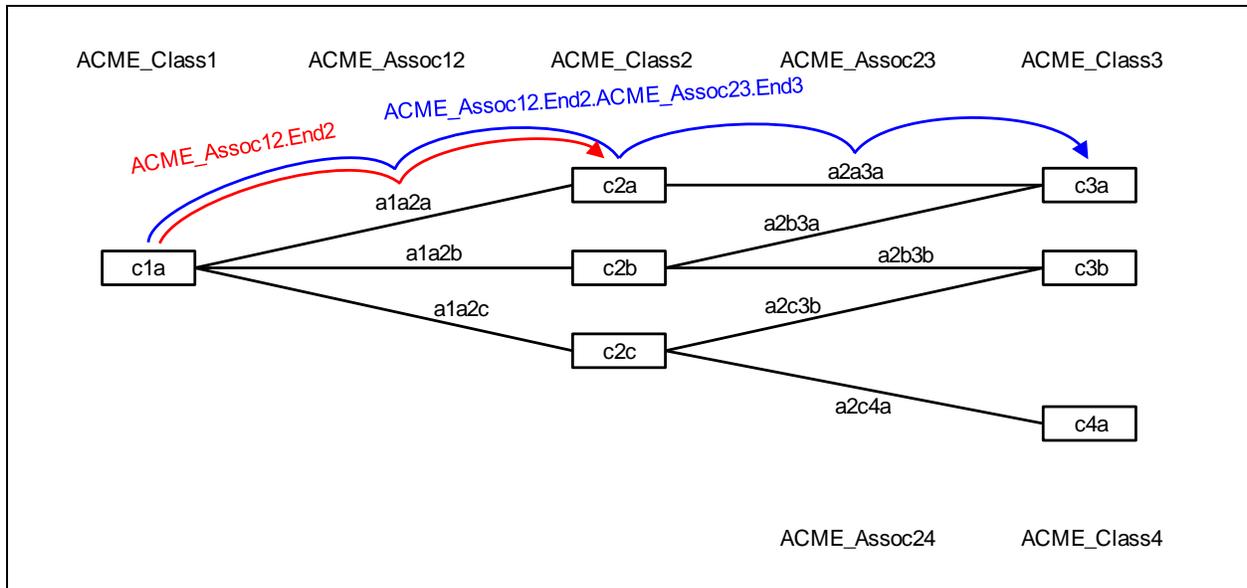
```

3799

3800 Note that the intermediate instances of class ACME_Class2 do not show up in the result. Some of them
 3801 are being traversed in the course of getting to the result instances, but because only the end result is
 3802 represented, the navigation path to get there does not show up.

3803 **D.1.7 Navigation with two paths that form a subset (merge)**

3804 In this example, the client retrieves an instance and specifies two navigation path: one that identifies
 3805 instances directly associated to the instance being retrieved, and one that identifies instances associated
 3806 across one additional association hop. Figure D-7 shows the instance diagram and the two navigation
 3807 paths, in blue and red. The red one is a subset of the blue one, so that they can be merged if the red one
 3808 is used with \$expand.



3809
 3810

3811 **Figure D-7 – Example instance diagram for navigation with two paths that form a subset (merge)**

3812 An instance retrieval request using these two navigation paths with the \$refer query parameter will return
 3813 the following instance representation:

```

3814 GET /c1a?$refer=ACME_Assoc12.End2,ACME_Assoc12.End2.ACME_Assoc23.End3
3815
3816 Instance c1a:
3817   Prop1: "...
3818   ACME_Assoc12.End2: ReferenceCollection:
3819     ref c2a
3820     ref c2b
3821     ref c2c
3822   ACME_Assoc12.End2.ACME_Assoc23.End3: ReferenceCollection:
3823     ref c3a
3824     ref c3a
3825     ref c3b
3826     ref c3b
    
```

3827 Note that the two navigation properties have not been merged, even though one navigation path was a
3828 subset of the other. The reason is that the shorter one was not expanded to instances.

3829 A changed request where the shorter navigation path is used with the \$expand query parameter and the
3830 longer one is used with \$refer will return the following instance representation:

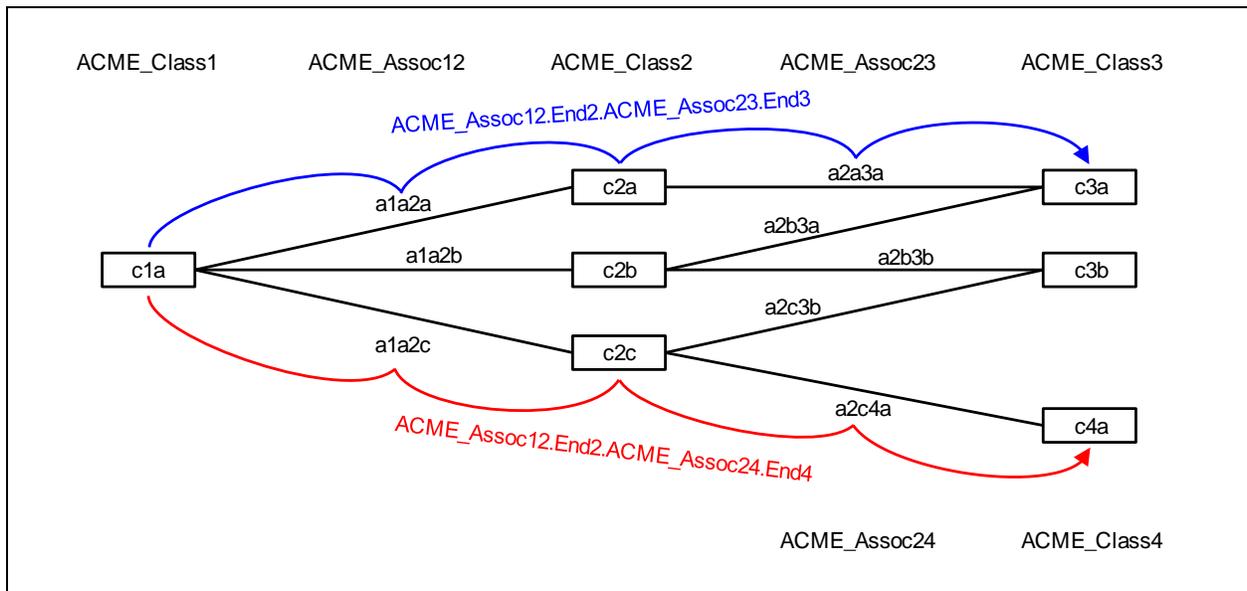
```
3831 GET /c1a?$expand=ACME_Assoc12.End2&$refer=ACME_Assoc12.End2.ACME_Assoc23.End3
3832
3833 Instance c1a:
3834     Prop1: "..."/>

```

3849 Note that the two navigation properties now have been merged, and that the names of the inner
 3850 navigation properties are relative to their starting point (that is, just "ACME_Assoc23.End3" and not
 3851 "ACME_Assoc12.End2.ACME_Assoc23.End3" as specified in the query parameter).

3852 **D.1.8 Navigation with two paths that have a common begin**

3853 This example is similar to the previous one, except that the two navigation paths have a common path
 3854 after their start but none is a subset of the other. Figure D-8 shows the instance diagram and the two
 3855 navigation paths, in blue and red.



3856
3857

3858 **Figure D-8 – Example instance diagram for navigation with two paths that have a common begin**

3859 An instance retrieval request using these two navigation paths with the \$refer query parameter will again
 3860 return an instance representation with two unmerged navigation properties; it is not shown for brevity.

3861 An instance retrieval request using one of these navigation paths with the \$expand query parameter will
 3862 also return an instance representation with two unmerged navigation properties:

```
3863 GET /c1a?$expand=ACME_Assoc12.End2.ACME_Assoc23.End3&$refer=ACME_Assoc12.End2.ACME_Assoc24.End4
3864
3865 Instance c1a:
3866   Prop1: "...
3867   ACME_Assoc12.End2.ACME_Assoc23.End3: InstanceCollection:
3868     Instance c3a:
3869       Prop3: "...
3870     Instance c3a:
3871       Prop3: "...
3872     Instance c3b:
3873       Prop3: "...
3874     Instance c3b:
3875       Prop3: "...
3876     Instance c3b:
3877       Prop3: "...
3877   ACME_Assoc12.End2.ACME_Assoc24.End4: ReferenceCollection:
```

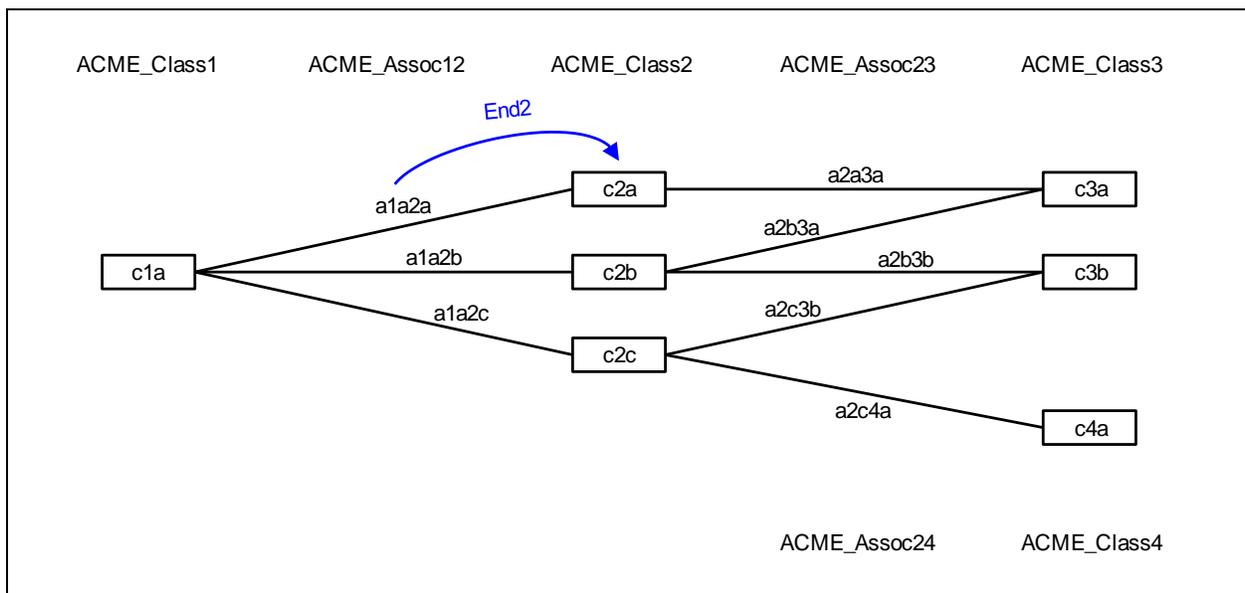
3878 ref c4a

3879 The reason for not merging is that the second property would need to have an anchor point for merging
 3880 (for example, ACME_Class2 instances), and such an anchor point is not provided by the first property,
 3881 because it only represents its end of the navigation path (instances referenced by End3).

3882 This does not change even when both navigation paths are expanded, because either result is just
 3883 representing the end of the navigation without providing an anchor point for the other.

3884 D.1.9 Expansion of association reference

3885 In this example, the client retrieves an association instance and specifies a navigation path that expands
 3886 one of the existing references in the association. Figure D-9 shows the instance diagram and the blue
 3887 navigation path "End2", starting at instance a1a2a.



3888
 3889

3890 **Figure D-9 – Example instance diagram for expansion of association reference**

3891 An instance retrieval request using this navigation path with the \$expand query parameter will return the
 3892 following instance representation:

3893 GET /a1a2a?\$expand=End2

3894

3895 Instance a1a2a:

3896 Prop12: "..."

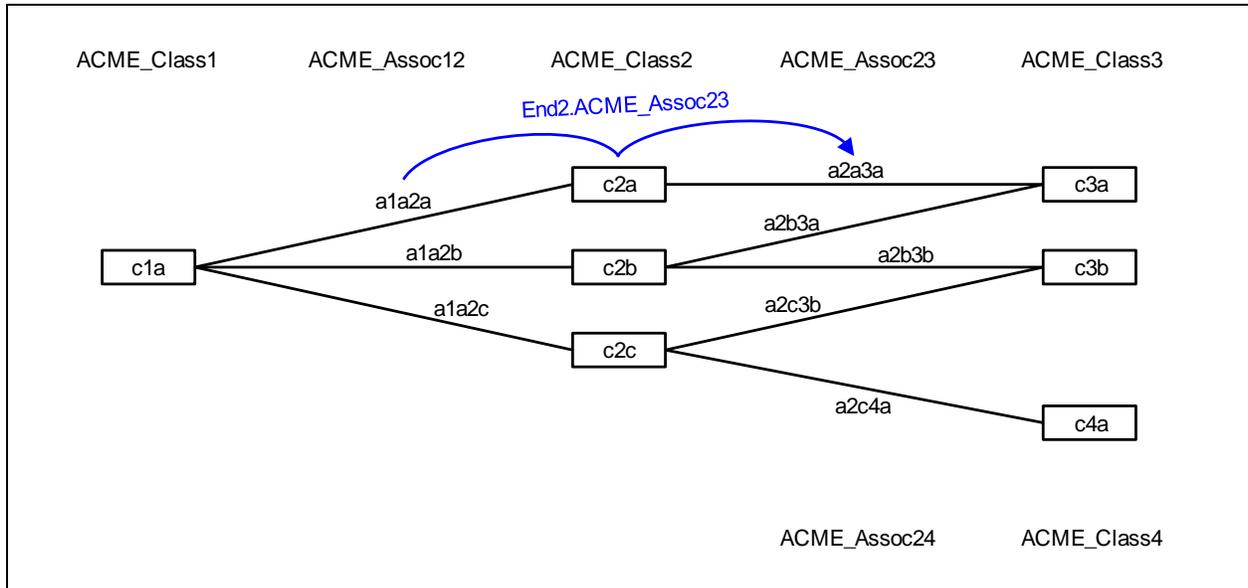
3897 End1: ref c1a

3898 End2: Instance c2a:

3899 Prop2: "..."

3900 **D.1.10 Navigation from association to referencing association**

3901 In this example, the client retrieves an association instance and specifies a navigation path that identifies
 3902 the association instances that reference the same instances that are also referenced by the association
 3903 instance being retrieved. Figure D-10 shows the instance diagram and the blue navigation path
 3904 "End2.ACME_Assoc23", starting at instance a1a2a.



3905
3906

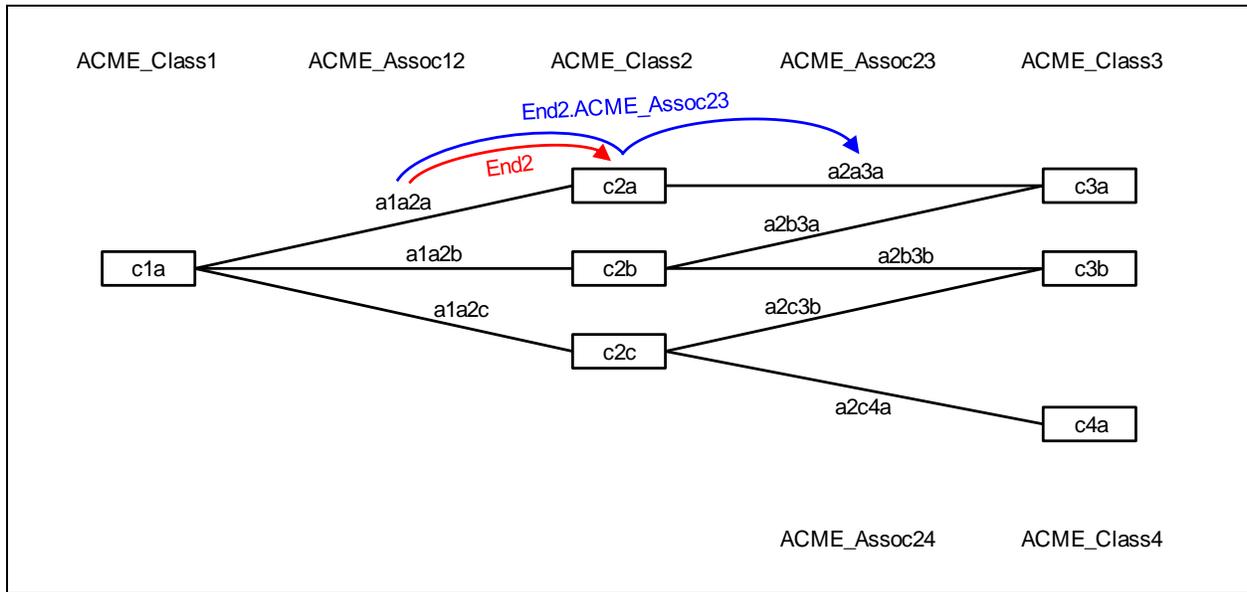
3907 **Figure D-10 – Example instance diagram for navigation starting from association**

3908 An instance retrieval request using this navigation path with the \$expand query parameter will return the
 3909 following instance representation:

```
3910 GET /a1a2a?$expand=End2.ACME_Assoc12
3911
3912 Instance a1a2a:
3913   Prop12: "...
3914   End1: ref c1a
3915   End2: ref c2a
3916   End2.ACME_Assoc12: InstanceCollection:
3917     Instance a2a3a:
3918       Prop23: "...
3919       End2: ref c2a
3920       End3: ref c3a
```

3921 **D.1.11 Expansion of association reference and navigation to referencing association**
 3922 **(merge)**

3923 In this example, the client retrieves an association instance and specifies both navigation properties from
 3924 the previous two examples. Figure D-11 shows the instance diagram, the red navigation path "End2", and
 3925 the blue navigation path "End2.ACME_Assoc23", both starting at instance a1a2a.



3926
 3927

3928 **Figure D-11 – Example instance diagram for expansion of association reference and navigation to**
 3929 **referencing association (merge)**

3930 An instance retrieval request using these navigation paths with the \$expand query parameter will return
 3931 the following instance representation:

```

3932 GET /a1a2a?$expand=End2,End2.ACME_Assoc12
3933
3934 Instance a1a2a:
3935   Prop12: "..."/>

```

3944 The two navigation paths get merged because one is a subset of the other. The inner navigation property
 3945 (specified using the navigation path "End2.ACME_Assoc12") gets merged into the existing reference
 3946 "End2" and its name gets shortened to "ACME_Assoc12" because that would be the valid navigation path
 3947 in the context of instance c2a.

3948 D.2 Paged retrieval

3949 This annex provides an example for paged retrieval, as described in 7.3.8. The example is based on the
 3950 classes defined in D.1 and assumes that the client has specified a maximum size for pageable collections
 3951 of 2 by using the \$max parameter (see 6.5.5), in order to demonstrate paging with a small number of
 3952 entities.

3953 Because the information that controls paging is represented in the payload, the requests and responses
 3954 are shown in detail instead of using the abbreviated notation used in D.1.

3955 D.2.1 Navigation to associated instances

3956 The following exchange shows the example from D.1.3 that includes a navigation property with
 3957 references to associated instances.

3958 Request:

```
3959 GET /cimrs/root%2Fcimv2/ACME_Class1/c1a?$refer=ACME_Assoc12.End2&$max=2 HTTP/1.1
3960 Host: server.acme.com:5988
3961 Accept: application/json;version=1.0
3962 X-CIMRS-Version: 1.0.0
```

3963 Response:

```
3964 HTTP/1.1 200 OK
3965 Date: Fri, 11 Nov 2011 10:11:00 GMT
3966 Content-Length: XXX
3967 Content-Type: application/json;version=1.0.1
3968 X-CIMRS-Version: 1.0.1
3969
3970 {
3971   "kind": "instance",
3972   "self": "/cimrs/root%2Fcimv2/ACME_Class1/c1a",
3973   "class": "ACME_Class1",
3974   "properties": {
3975     "Prop1": "...",
3976     "ACME_Assoc12.End2": {
3977       "kind": "referencecollection",
3978       "self": "/cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/1",
3979       "next": "/cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/2",
3980       "class": "ACME_Class2",
3981       "references": [
3982         "/cimrs/root%2Fcimv2/ACME_Class2/c2a",
3983         "/cimrs/root%2Fcimv2/ACME_Class2/c2b"
3984       ]
3985     }
3986   },
3987   "methods": { ... }
3988 }
```

3989 The presence of the "next" attribute in the reference collection indicates that there are more pages to
 3990 retrieve, so the client issues a request to retrieve the next page of that collection:

3991 Request:

```
3992 GET /cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/2?$max=2
3993 HTTP/1.1
3994 Host: server.acme.com:5988
3995 Accept: application/json;version=1.0
3996 X-CIMRS-Version: 1.0.0
```

3997 Response:

```
3998 HTTP/1.1 200 OK
3999 Date: Fri, 11 Nov 2011 10:11:00 GMT
4000 Content-Length: XXX
4001 Content-Type: application/json;version=1.0.1
4002 X-CIMRS-Version: 1.0.1
4003
4004 {
4005     "kind": "referencecollection",
4006     "self": "/cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/2",
4007     "class": "ACME_Class2",
4008     "references": [
4009         "/cimrs/root%2Fcimv2/ACME_Class2/c2c"
4010     ]
4011 }
```

4012 This time, the reference collection does not contain a next attribute, indicating that the collection is now
4013 complete.

4014 The variant using the \$expand parameter is omitted; paged retrieval works the same for that variant
4015 except that the response now contains an instance collection instead of the reference collection. See
4016 7.8.2 for an example of an instance collection retrieval.

4017
4018
4019
4020

ANNEX E (informative)

Change log

Version	Date	Description
1.0.0	2013-01-24	

Bibliography

- 4021
- 4022 This annex contains a list of non-normative references for this document.
- 4023 DMTF DSP0200, *CIM Operations over HTTP 1.3*,
4024 http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf
- 4025 DMTF DSP1001, *Management Profile Specification Usage Guide 1.1*,
4026 http://www.dmtf.org/standards/published_documents/DSP1001_1.1.pdf
- 4027 DMTF DSP1033, *Profile Registration Profile 1.0*,
4028 http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf
- 4029 DMTF DSP1054, *Indications Profile 1.2*,
4030 http://www.dmtf.org/sites/default/files/standards/documents/DSP1054_1.2.pdf
- 4031 DMTF DSP2032, *CIM-RS White Paper 1.0*,
4032 http://www.dmtf.org/standards/published_documents/DSP2032_1.0.pdf
- 4033 ECMA-262, *ECMAScript Language Specification, 5th Edition*, December 2009,
4034 <http://www.ecma-international.org/publications/standards/Ecma-262.htm>
- 4035 IETF RFC2608, *Service Location Protocol, Version 2*, June 1999,
4036 <http://tools.ietf.org/html/rfc2608>
- 4037 IETF RFC4648, *The Base16, Base32, and Base64 Data Encodings*, October 2006,
4038 <http://tools.ietf.org/html/rfc4648>
- 4039 IETF RFC5005, *Feed Paging and Archiving*, September 2007,
4040 <http://tools.ietf.org/html/rfc5005>
- 4041 IETF Draft RFC *Additional HTTP Status Codes*, Draft 04, February 2012,
4042 <http://tools.ietf.org/html/draft-nottingham-http-new-status-04>
- 4043 IANA Permanent Message Header Field Names,
4044 <http://www.iana.org/assignments/message-headers/perm-headers.html>
- 4045 IANA MIME Media Types,
4046 <http://www.iana.org/assignments/media-types/>
- 4047 ITU-T X.509, *Information technology – Open Systems Interconnection – The Directory: Public-key and*
4048 *attribute certificate frameworks*,
4049 <http://www.itu.int/rec/T-REC-X.509/en>
- 4050 R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, PhD thesis,
4051 University of California, Irvine, 2000,
4052 <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- 4053 R. Fielding, *REST APIs must be hypertext driven*, October 2008,
4054 <http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven>
- 4055 J. Holzer, *RESTful Web Services and JSON for WBEM Operations*, Master thesis, University of Applied
4056 Sciences, Konstanz, Germany, June 2009,
4057 <http://mond.htwg-konstanz.de/Abschlussarbeiten/Details.aspx?id=1120>
- 4058 A. Manes, *Rest principle: Separation of representation and resource*, March 2009,
4059 <http://apsblog.burtongroup.com/2009/03/rest-principle-separation-of-representation-and-resource.html>

4060 L. Richardson and S. Ruby, *RESTful Web Services*, May 2007, O'Reilly, ISBN 978-0-596-52926-0,
4061 <http://www.oreilly.de/catalog/9780596529260/>