



1
2
3
4

Document Number: DSP0210

Date: 2014-02-11

Version: 1.0.1

5 **CIM-RS Protocol**

6 **Document Type: Specification**
7 **Document Status: DMTF Standard**
8 **Document Language: en-US**
9

10 Copyright Notice

11 Copyright © 2010–2014 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

CONTENTS

33	Foreword	8
34	Introduction.....	9
35	Document conventions.....	9
36	1 Scope	11
37	2 Normative references.....	11
38	3 Terms and definitions	12
39	4 Symbols and abbreviated terms.....	17
40	5 Concepts	18
41	5.1 CIM-RS protocol participants.....	18
42	5.2 Model independence of CIM-RS.....	19
43	5.3 Basic kinds of resources.....	21
44	5.4 Mapping model elements to CIM-RS resources.....	21
45	5.4.1 Classes	21
46	5.4.2 Instances.....	21
47	5.4.3 Properties.....	21
48	5.4.4 Methods and operations	22
49	5.5 Two-staged mapping approach	22
50	5.6 Navigation between resources (EXPERIMENTAL)	23
51	5.7 Discovering resources in a server	26
52	5.8 REST architectural style supported by CIM-RS	27
53	6 Resource identifiers.....	28
54	6.1 CIM-RS resource identifier format	28
55	6.2 Opacity.....	29
56	6.3 Percent-encoding.....	29
57	6.4 Authority component.....	31
58	6.5 Query parameters	31
59	6.5.1 \$class (specify class name).....	33
60	6.5.2 \$continueonerror (continue on errors within paged retrieval)	33
61	6.5.3 \$expand (include target instances, EXPERIMENTAL).....	34
62	6.5.4 \$filter (filter instances in result)	35
63	6.5.5 \$max (limit number of collection members in result)	36
64	6.5.6 \$methods (subset method links in result)	37
65	6.5.7 \$pageingtimeout (specify inactivity timeout for paged retrieval).....	38
66	6.5.8 \$properties (subset properties in result)	39
67	6.5.9 \$refer (include references to target instances, EXPERIMENTAL)	40
68	6.6 Resource identifiers of entry point resources	41
69	7 Resources, operations and payload elements	42
70	7.1 Overview	42
71	7.2 Description conventions.....	44
72	7.2.1 Datatypes used in payload element definitions	44
73	7.2.2 Requirement levels used in payload element definitions.....	45
74	7.2.3 Requirement levels used in operation definitions.....	45
75	7.2.4 CIM-RS operation description format	45
76	7.3 Common behaviors for all operations.....	46
77	7.3.1 Content negotiation.....	46
78	7.3.2 Verifying the basis of resource modifications (EXPERIMENTAL)	46
79	7.3.3 Caching of responses	46
80	7.3.4 Success and failure.....	47
81	7.3.5 Errors	47
82	7.3.6 ErrorResponse payload element	47
83	7.3.7 Consistency model.....	47

84	7.3.8	Paging of collections	48
85	7.4	Optional features of the CIM-RS protocol.....	50
86	7.4.1	Entity tagging feature	50
87	7.4.2	Continue on error feature.....	50
88	7.5	Instance creation resource	50
89	7.5.1	POST	50
90	7.6	Instance resource	52
91	7.6.1	Instance payload element.....	53
92	7.6.2	DELETE	53
93	7.6.3	GET	54
94	7.6.4	PUT	56
95	7.7	Reference collection resource	58
96	7.7.1	ReferenceCollection payload element	59
97	7.7.2	GET	59
98	7.8	Instance collection resource	61
99	7.8.1	InstanceCollection payload element	61
100	7.8.2	GET	61
101	7.9	Instance enumeration resource	64
102	7.9.1	GET	64
103	7.10	Method invocation resource.....	66
104	7.10.1	MethodRequest payload element	66
105	7.10.2	MethodResponse payload element	66
106	7.10.3	POST	67
107	7.11	Listener destination resource.....	69
108	7.11.1	IndicationDeliveryRequest payload element	69
109	7.11.2	POST	69
110	7.12	Server entry point resource	71
111	7.12.1	ServerEntryPoint payload element	71
112	7.12.2	GET	72
113	7.13	Listener entry point resource	74
114	7.13.1	ListenerEntryPoint payload element	74
115	7.13.2	GET	75
116	7.14	CIM-RS resources to be exposed.....	76
117	7.14.1	Resources exposed by a server	76
118	7.14.2	Resources exposed by a listener.....	77
119	7.15	Other typical WBEM protocol functionality.....	77
120	7.15.1	Server discovery	77
121	7.15.2	Discovery of server and listener entry point resources.....	77
122	7.15.3	Namespace discovery	77
123	7.15.4	Registered profile discovery	77
124	7.15.5	Schema inspection.....	78
125	7.15.6	Association traversal (EXPERIMENTAL)	78
126	7.15.7	Indication subscription	78
127	8	HTTP usage	78
128	8.1	General requirements	78
129	8.2	Authentication requirements	78
130	8.2.1	Operating without authentication	79
131	8.2.2	HTTP basic authentication.....	79
132	8.2.3	HTTP digest authentication	79
133	8.2.4	Other authentication mechanisms	79
134	8.3	Message encryption requirements.....	79
135	8.4	HTTP header fields	80
136	8.4.1	Accept	80
137	8.4.2	Content-Type	81
138	8.4.3	ETag (EXPERIMENTAL)	82

139 8.4.4 If-Match (EXPERIMENTAL)..... 82

140 8.4.5 X-CIMRS-Version 82

141 9 Payload representation 83

142 9.1 Internet media types 83

143 9.1.1 General 83

144 9.1.2 Media type parameters 83

145 9.2 Payload element representations 84

146 9.3 Payload representations 85

147 10 Discovery requirements..... 85

148 11 Version compatibility 85

149 11.1 HTTP protocol version compatibility 85

150 11.2 CIM-RS protocol version compatibility 86

151 11.3 CIM-RS payload representation version compatibility 86

152 12 Conformance..... 86

153 ANNEX A (normative) Common ABNF rules 87

154 ANNEX B (informative) Mapping CIM-RS to generic operations 88

155 B.1 URI composition..... 88

156 B.1.1 Instance creation resource 88

157 B.1.2 Instance resource 88

158 B.1.3 Page of instance or reference collection resource from association traversal
159 (EXPERIMENTAL)..... 88

160 B.1.4 Page of instance or reference collection resource from enumeration by class 89

161 B.1.5 Instance enumeration resource 89

162 B.1.6 Static method invocation resource..... 89

163 B.1.7 Non-static method invocation resource 90

164 B.1.8 Listener destination resource..... 90

165 B.1.9 Server and listener entry point resources 90

166 B.2 Query parameters 90

167 B.2.1 Special handling for \$expand and \$refer query parameters (EXPERIMENTAL) 91

168 B.3 Server operations..... 93

169 B.3.1 POST instance creation resource..... 93

170 B.3.2 POST static method invocation resource 93

171 B.3.3 POST non-static method invocation resource 94

172 B.3.4 DELETE instance resource 94

173 B.3.5 GET instance resource 95

174 B.3.6 GET page of instance collection resource 95

175 B.3.7 GET page of reference collection resource 96

176 B.3.8 GET instance enumeration resource 96

177 B.3.9 GET server entry point resource..... 97

178 B.3.10 PUT instance resource 98

179 B.4 Listener operations 98

180 B.4.1 POST listener destination resource 98

181 B.4.2 GET listener entry point resource 99

182 ANNEX C (informative) Mapping generic operations to CIM-RS..... 100

183 C.1 Conformance 100

184 C.2 Support of optional generic operations features 100

185 C.3 Operations supported 100

186 C.3.1 GetInstance..... 100

187 C.3.2 DeleteInstance 101

188 C.3.3 ModifyInstance..... 101

189 C.3.4 CreateInstance..... 101

190 C.3.5 OpenEnumerateInstances 101

191 C.3.6 OpenEnumerateInstancePaths..... 102

192 C.3.7 OpenAssociators (EXPERIMENTAL) 102

193 C.3.8 OpenAssociatorPaths (EXPERIMENTAL)..... 103

194 C.3.9 OpenReferences (EXPERIMENTAL) 104

195 C.3.10 OpenReferencePaths (EXPERIMENTAL) 105

196 C.3.11 PullInstancesWithPath 106

197 C.3.12 PullInstancePaths 107

198 C.3.13 InvokeMethod 107

199 C.3.14 InvokeStaticMethod 107

200 C.4 Operations not supported 108

201 C.4.1 Direct instance enumeration operations 108

202 C.4.2 Class and qualifier type operations 108

203 C.4.3 Other operations 109

204 ANNEX D (informative) Examples 110

205 D.1 Navigation between resources (EXPERIMENTAL) 110

206 D.1.1 Classes and instances used in the examples 110

207 D.1.2 Navigation to referencing association instances 112

208 D.1.3 Navigation to associated instances 113

209 D.1.4 Navigation to association instances across one hop 114

210 D.1.5 Navigation to associated instances across two hops 116

211 D.1.6 Navigation to associated instances across two hops (2) 117

212 D.1.7 Navigation with two paths that form a subset (merge) 118

213 D.1.8 Navigation with two paths that have a common begin 120

214 D.1.9 Expansion of association reference 121

215 D.1.10 Navigation from association to referencing association 122

216 D.1.11 Expansion of association reference and navigation to referencing association

217 (merge) 123

218 D.2 Paged retrieval 124

219 D.2.1 Navigation to associated instances (EXPERIMENTAL) 124

220 ANNEX E (informative) Change log 126

221 Bibliography 127

222 **Figures**

223 Figure 1 – Participants in the CIM-RS protocol..... 19

224 Figure 2 – Single model and multiple protocols 20

225 Figure 3 – Two-staged mapping approach in CIM-RS 23

226 Figure 4 – Expanding association classes to construct navigation paths 25

227 Figure D-1 – Class diagram for navigation examples 110

228 Figure D-2 – Example instance diagram for navigation to referencing association instances 112

229 Figure D-3 – Example instance diagram for navigation to associated instances 113

230 Figure D-4 – Example instance diagram for navigation to association instances across one hop 114

231 Figure D-5 – Example instance diagram for navigation to associated instances across two hops 116

232 Figure D-6 – Example instance diagram for navigation to associated instances across two hops (2)..... 117

233 Figure D-7 – Example instance diagram for navigation with two paths that form a subset (merge) 118

234 Figure D-8 – Example instance diagram for navigation with two paths that have a common begin 120

235 Figure D-9 – Example instance diagram for expansion of association reference 121

236 Figure D-10 – Example instance diagram for navigation starting from association..... 122

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

238 referencing association (merge) 123

239 **Tables**

240

241 Table 1 – Query parameters in CIM-RS 32

242 Table 2 – Resource types in CIM-RS 42

243 Table 3 – CIM-RS operations..... 43

244 Table 4 – CIM-RS payload elements 43

245 Table 5 – Datatypes used in payload elements 44

246 Table 6 – Attributes of an ErrorResponse payload element 47

247 Table 7 – Operations supporting paging of collections 49

248 Table 8 – Attributes of an Instance payload element..... 53

249 Table 9 – Attributes of an ReferenceCollection payload element..... 59

250 Table 10 – Attributes of an InstanceCollection payload element..... 61

251 Table 11 – Attributes of a MethodRequest payload element..... 66

252 Table 12 – Attributes of a MethodResponse payload element 67

253 Table 13 – Attributes of an IndicationDeliveryRequest payload element 69

254 Table 14 – Attributes of a ServerEntryPoint payload element 71

255 Table 15 – Attributes of SEPNamespace payload datatype 72

256 Table 16 – Attributes of a ListenerEntryPoint payload element..... 74

257 Table 17 – Media type parameters 83

258 Table 18 – CIM-RS payload representations 85

259 Table B-1 – Mapping of CIM-RS query parameters to generic operations input parameters 90

260 Table C-1 – Pulled equivalents of direct instance enumeration operations 108

261

262

Foreword

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

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

267 Acknowledgments

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

- 269 • Cornelia Davis, EMC
- 270 • George Ericson, EMC
- 271 • Johannes Holzer, IBM
- 272 • Robert Kieninger, IBM
- 273 • Wojtek Kozaczynski, Microsoft
- 274 • Larry Lamers, VMware
- 275 • Andreas Maier, IBM (editor)
- 276 • Bob Tillman, EMC
- 277 • Marvin Waschke, CA Technologies

278

Introduction

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

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

285 Document conventions

286 Typographical conventions

287 The following typographical conventions are used in this document:

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

290 ABNF usage conventions

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

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

300 The following general ABNF rules are defined:

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

302 Experimental material

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

308 The following typographical convention indicates experimental material:

309 EXPERIMENTAL

310 Experimental material appears here.

311 EXPERIMENTAL

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

314

316

CIM-RS Protocol

317 1 Scope

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

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

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

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

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

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

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

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

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

341 2 Normative references

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

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

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

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

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

- 354 DMTF DSP0223, *Generic Operations 1.0*,
355 http://www.dmtf.org/standards/published_documents/DSP0223_1.0.pdf
- 356 DMTF DSP0211, *CIM-RS Payload Representation in JSON 1.0*,
357 http://www.dmtf.org/standards/published_documents/DSP0211_1.0.pdf
- 358 IETF RFC2246, *The TLS Protocol Version 1.0*, January 1999,
359 <http://tools.ietf.org/html/rfc2246>
- 360 IETF RFC2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999,
361 <http://tools.ietf.org/html/rfc2616>
- 362 IETF RFC2617, *HTTP Authentication: Basic and Digest Access Authentication*, June 1999,
363 <http://tools.ietf.org/html/rfc2617>
- 364 IETF RFC2818, *HTTP Over TLS*, May 2000,
365 <http://tools.ietf.org/html/rfc2818>
- 366 IETF RFC3986, *Uniform Resource Identifier (URI): Generic Syntax*, January 2005,
367 <http://tools.ietf.org/html/rfc3986>
- 368 IETF RFC4346, *The Transport Layer Security (TLS) Protocol, Version 1.1*, April 2006,
369 <http://tools.ietf.org/html/rfc4346>
- 370 IETF RFC5234, *Augmented BNF for Syntax Specifications: ABNF*, January 2008,
371 <http://tools.ietf.org/html/rfc5234>
- 372 IETF RFC5246, *The Transport Layer Security (TLS) Protocol, Version 1.2*, August 2008,
373 <http://tools.ietf.org/html/rfc5246>
- 374 ISO/IEC 10646:2003, *Information technology -- Universal Multiple-Octet Coded Character Set (UCS)*,
375 [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)
- 376 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards (2004, 5th*
377 *edition)*,
378 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse>
- 379 NIST Special Publication 800-57, Elaine Barker et al, *Recommendation for Key Management – Part 1:*
380 *General (Revised)*, March 2007,
381 http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf
- 382 NIST Special Publication 800-131A, Elaine Barker and Allen Roginsky, *Transitions: Recommendation for*
383 *Transitioning the Use of Cryptographic Algorithms and Key Lengths*, January 2011,
384 <http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf>
- 385 The Unicode Consortium, *The Unicode Standard, Version 5.2.0, Annex #15: Unicode Normalization*
386 *Forms*,
387 <http://www.unicode.org/reports/tr15/>

388 **3 Terms and definitions**

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

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

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

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

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

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

406 The following additional terms are used in this document.

407 **3.1**

408 **CIM-RS operation**

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

411 **3.2**

412 **CIM-RS payload element**

413 a particular type of content of the entity body of the HTTP messages used by the CIM-RS protocol.
414 Payload elements are abstractly defined in this document, and concretely in CIM-RS payload
415 representation specifications. For the list of payload elements defined for the CIM-RS protocol, see Table
416 4.

417 **3.3**

418 **CIM-RS payload representation**

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

422 **3.4**

423 **CIM-RS payload representation specification**

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

425 **3.5**

426 **CIM-RS protocol**

427 the protocol defined in this document and related documents.

428 **3.6**

429 **CIM-RS resource**

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

433 **3.7**

434 **CIM-RS resource identifier**

435 a URI that is a reference to a CIM-RS resource in a WBEM server or WBEM listener, as defined in 6. Also
436 called "resource identifier" in this document.

- 437 **3.8**
438 **HTTP basic authentication**
439 a simple authentication scheme for use by HTTP and HTTPS that is based on providing credentials in
440 HTTP header fields. It is defined in [RFC2617](#).
- 441 **3.9**
442 **HTTP content negotiation**
443 a method for selecting a representation of content in an HTTP response message when there are multiple
444 representations available. It is defined in section 12 of [RFC2616](#). Its use in the CIM-RS protocol is
445 described in 7.3.1.
- 446 **3.10**
447 **HTTP digest authentication**
448 an authentication scheme for use by HTTP and HTTPS that is based on verifying shared secrets that are
449 not exchanged. It is defined in [RFC2617](#).
- 450 **3.11**
451 **HTTP entity body**
452 the payload within an HTTP message, as defined in section 7.2 of [RFC2616](#).
- 453 **3.12**
454 **HTTP entity-header field**
455 a header field that may be used in HTTP requests and HTTP response messages, specifying information
456 that applies to the data in the entity body. Also called "HTTP entity-header".
- 457 **3.13**
458 **HTTP extension-header field**
459 an entity-header field used for custom extensions to the standard set of header fields defined in
460 [RFC2616](#). Also called "HTTP extension-header".
- 461 **3.14**
462 **HTTP general-header field**
463 a header field that may be used in HTTP requests and HTTP response messages, specifying information
464 that applies to the HTTP message. Also called "HTTP general-header".
- 465 **3.15**
466 **HTTP header field**
467 a named value used in the header of HTTP messages, as defined in section 4.2 of [RFC2616](#). Also called
468 "HTTP header". The specific types of header fields are general-header field, request-header field,
469 response-header field, entity-header field, and extension-header field.
- 470 **3.16**
471 **HTTP message**
472 an interaction between an HTTP client and an HTTP server (in any direction), as defined in section 4 of
473 [RFC2616](#).
- 474 **3.17**
475 **HTTP method**
476 the type of interaction stated in HTTP requests, as defined in section 5.1.1 of [RFC2616](#).

477 **3.18**478 **HTTP request message**

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

481 **3.19**482 **HTTP request-header field**

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

485 **3.20**486 **HTTP response message**

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

489 **3.21**490 **HTTP response-header field**

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

493 **3.22**494 **Internet media type**

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

498 **3.23**499 **Interop namespace**

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

503 **3.24**504 **method invocation link**

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

506 **3.25**507 **model**

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

511 **3.26**512 **navigation property**

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

515 **3.27**516 **Normalization Form C**

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

- 520 **3.28**
521 **reference-typed parameter**
522 a CIM method parameter declared with a CIM datatype that is a reference to a specific class.
- 523 **3.29**
524 **reference-typed property**
525 a CIM property declared with a CIM datatype that is a reference to a specific class. See 5.4.3 for details.
526 [DSP0004](#) defines the term "reference" for such properties; this document uses the more specific term
527 "reference-typed property", instead.
- 528 **3.30**
529 **reference-qualified property**
530 a string-typed CIM property qualified with the *Reference* qualifier (see [DSP0004](#) for a definition of the
531 *Reference* qualifier, and 5.4.3 for details).
- 532 **3.31**
533 **reference property**
534 a general term for reference-typed properties and reference-qualified properties. See 5.4.3 for details.
- 535 **3.32**
536 **resource representation**
537 a representation of a resource or some aspect thereof, in some format. A particular resource may have
538 any number of representations. The format of a resource representation is identified by a media type. In
539 the CIM-RS protocol, the more general term "payload representation" is used, because not all payload
540 elements are resource representations.
- 541 **3.33**
542 **REST architectural style**
543 the architectural style described in [Architectural Styles and the Design of Network-based Software](#)
544 [Architectures](#), chapter 5, and in [REST APIs must be hypertext driven](#).
- 545 **3.34**
546 **UCS character**
547 a character from the Universal Character Set defined in [ISO/IEC 10646:2003](#). See also [DSP0004](#) for the
548 usage of UCS characters in CIM strings. An alternative term is "Unicode character".
- 549 **3.35**
550 **WBEM client**
551 the client role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see 5.1.
- 552 **3.36**
553 **WBEM listener**
554 the event listener role in the CIM-RS protocol and in other WBEM protocols.. For a full definition, see 5.1.
- 555 **3.37**
556 **WBEM server**
557 the server role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see 5.1.

558 4 Symbols and abbreviated terms

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

561 4.1

562 ABNF

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

564 4.2

565 CIM

566 Common Information Model, as defined by DMTF.

567 4.3

568 CIM-RS

569 CIM RESTful Services

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

571 4.4

572 FQL

573 Filter Query Language, as defined by DMTF.

574 4.5

575 HTTP

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

578 4.6

579 HTTPS

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

581 4.7

582 IANA

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

584 4.8

585 JSON

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

587 4.9

588 REST

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

591 4.10

592 SLP

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

594 4.11

595 UCS

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

597 **4.12**
598 **URI**
599 Uniform Resource Identifier, as defined in [RFC3986](#).

600 **4.13**
601 **UTF-8**
602 UCS Transformation Format 8, as defined in [ISO/IEC 10646:2003](#).

603 **4.14**
604 **WBEM**
605 Web Based Enterprise Management, as defined by DMTF.

606 **4.15**
607 **XML**
608 eXtensible Markup Language, as defined by W3C.

609 **5 Concepts**

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

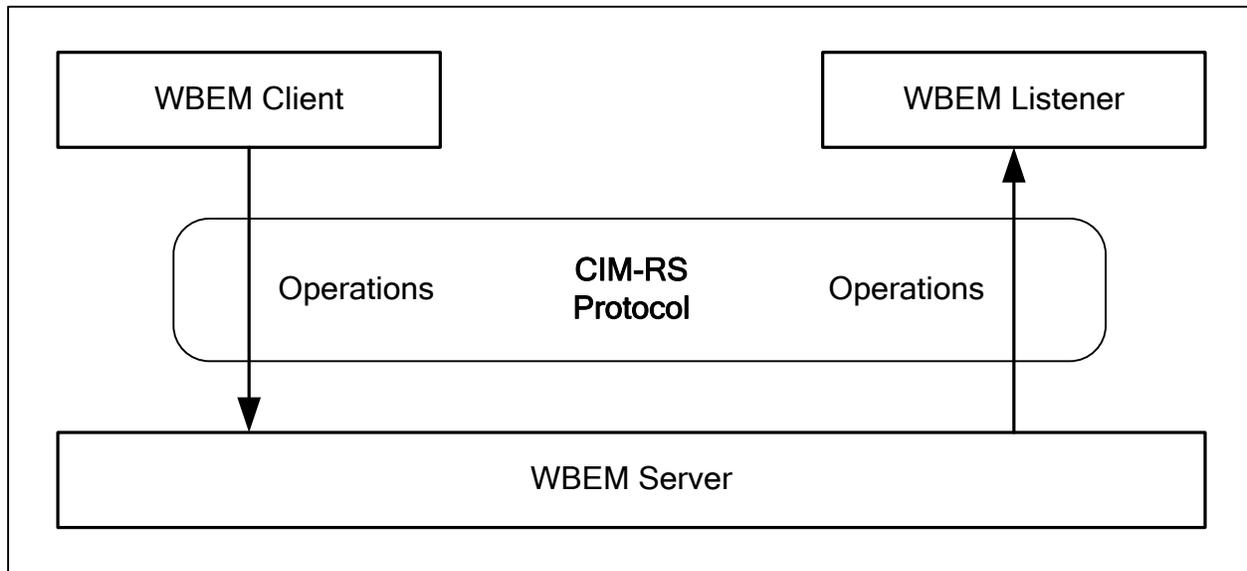
611 **5.1 CIM-RS protocol participants**

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

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

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

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



622
623

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

625 5.2 Model independence of CIM-RS

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

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

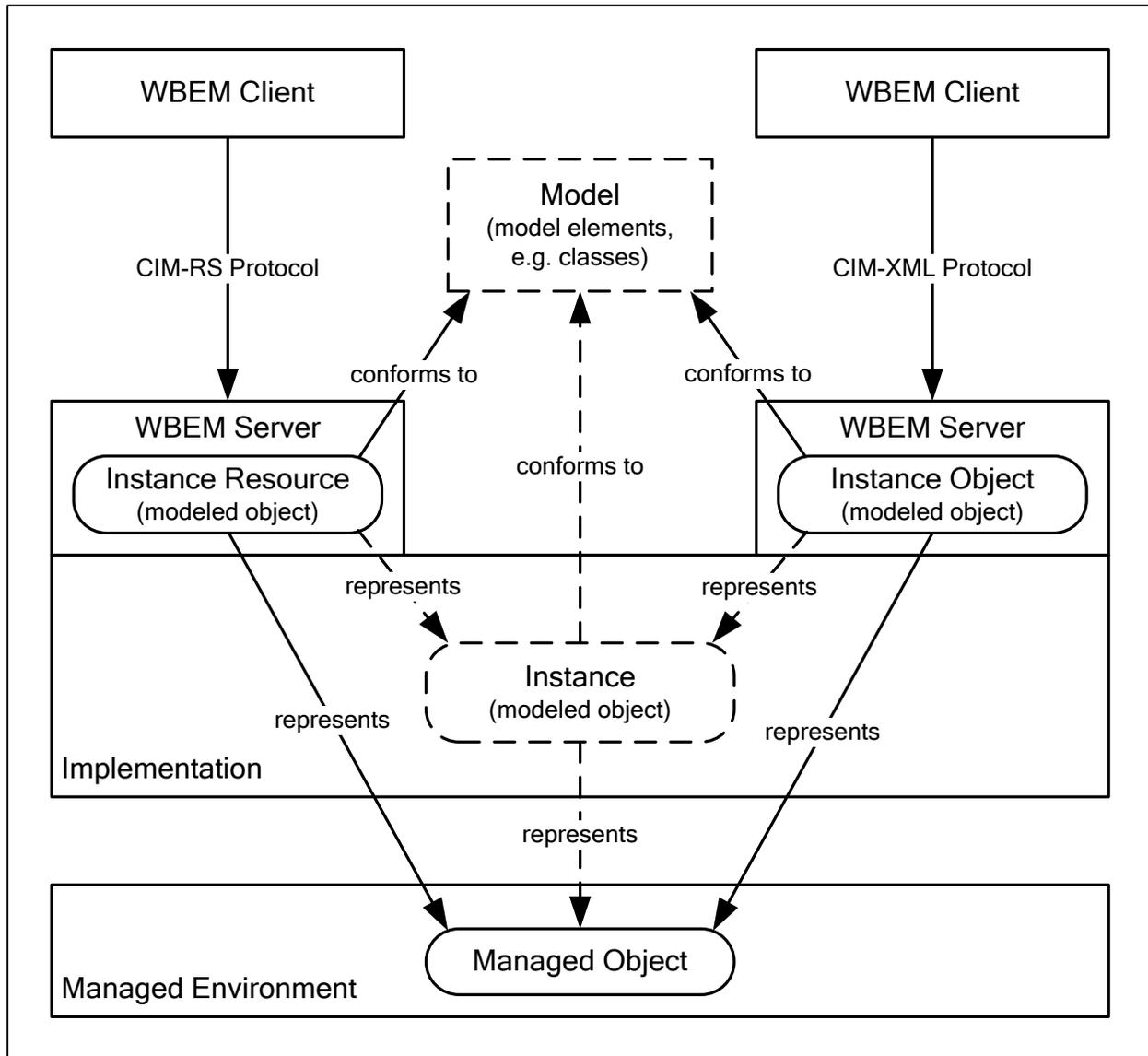
635 The CIM Schema published by DMTF is an example of a model that is conformant to [DSP0004](#), but any
636 [DSP0004](#) conformant model can be used with the CIM-RS protocol. Such other models are not required
637 to be derived from the CIM Schema published by DMTF. In this document, the term "model"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

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

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

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

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



661
662

Figure 2 – Single model and multiple protocols

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

668 5.3 Basic kinds of resources

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

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

675 5.4 Mapping model elements to CIM-RS resources

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

677 5.4.1 Classes

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

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

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

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

689 5.4.2 Instances

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

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

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

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

698 5.4.3 Properties

699 Properties of addressable instances are represented as properties of the corresponding instance
700 resources. Properties of instances that are not addressable are represented as properties of the
701 corresponding instances embedded in payload elements.

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

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

- 706 • reference-typed properties – These are reference properties in association classes that are
707 declared with a CIM datatype that is a reference to a specific class; they are the ends of
708 associations. Reference-typed properties are always scalars; there are no arrays of reference-
709 typed properties. The value of a reference-typed property references a single instance.
- 710 • reference-qualified properties – These are string-typed properties that are qualified with the
711 *Reference* qualifier. These properties can be used in ordinary classes; they are like simple
712 pointers to instances and do not constitute association ends or imply any associations.
713 Reference-qualified properties may be scalars or arrays. The value of a reference-qualified
714 scalar property and the value of an array entry of a reference-qualified array property reference
715 a single instance.

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

718 5.4.4 Methods and operations

719 Class-defined (extrinsic) methods can be defined as being static or non-static. Non-static methods that
720 are implemented are exposed via method invocation links in each instance (see 7.6). Static methods that
721 are implemented are exposed via method invocation links in the global server entry point resource (see
722 7.12). Details on method invocation links are defined in Table 5.

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

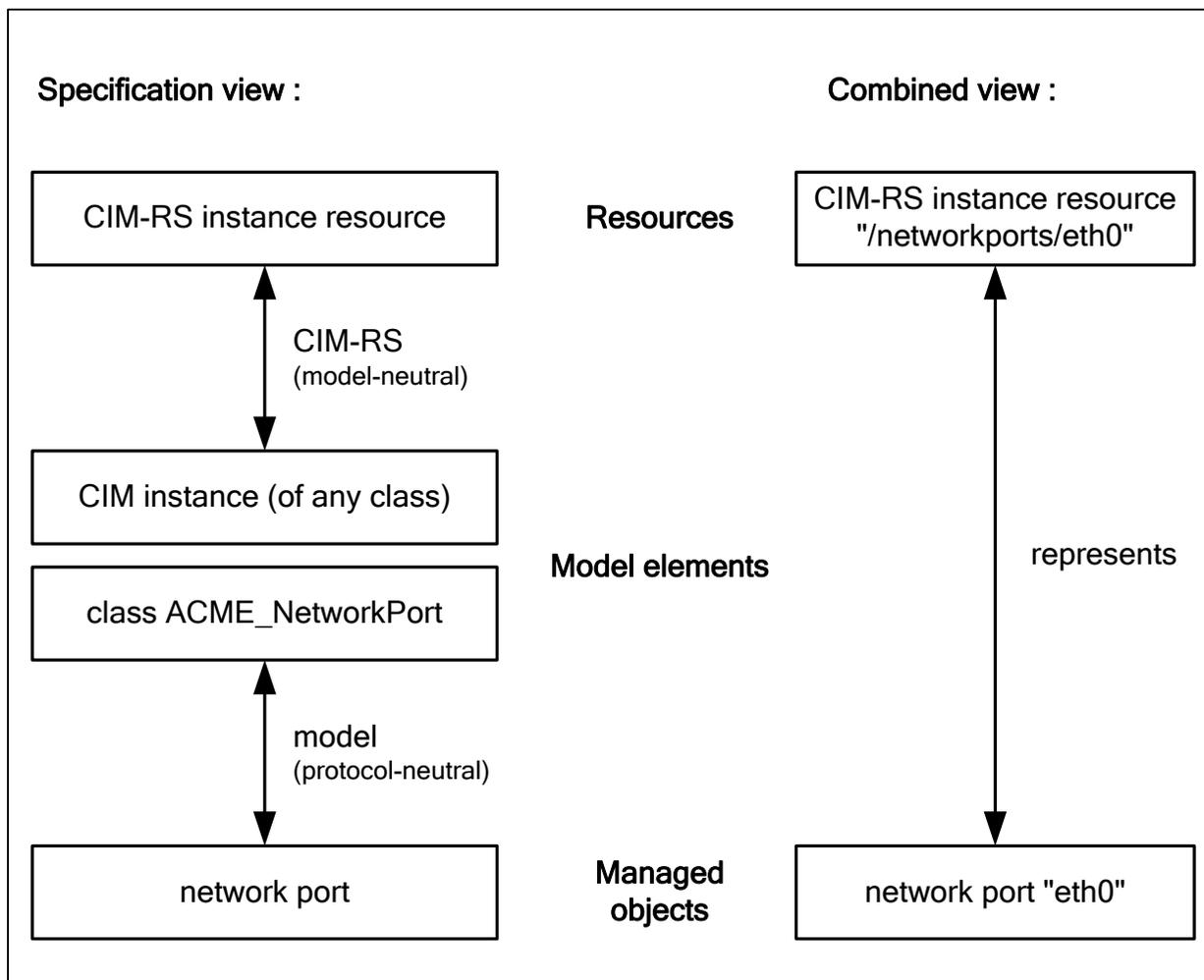
727 5.5 Two-staged mapping approach

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

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

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

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



736

737

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

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

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

742 **5.6 Navigation between resources (EXPERIMENTAL)**

743 **EXPERIMENTAL**

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

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

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

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

755 The values of the `$expand` and `$refer` query parameters are lists of navigation paths.

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

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

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

763 The value of navigation properties included due to the usage of `$refer` is a reference or collection of
764 references to these identified target instances, while the value of navigation properties included due to the
765 usage of `$expand` is the identified target instance or collection of target instances. For more details on
766 the values of navigation properties and on the query parameter syntax, see the descriptions of `$expand`
767 (see 6.5.3) and `$refer` (see 6.5.9).

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

```
769 nav-path = nav-hop *( "." nav-hop )
770
771 nav-hop = nav-filter ( embedded-path ref-name / assoc-class-name )
772
773 embedded-path = *( prop-name "." )
774
775 nav-filter = ( "[" filter-class-name "]" )
```

776 Where:

- 777 • `nav-hop` identifies a set of instances at the current hop, based on the instances at the previous
778 hop, as follows:
 - 779 – If `ref-name` is specified in `nav-hop`, `ref-name` shall either be the name of an existing
780 (typed or qualified) reference exposed by the instances at the current hop, or the name of a
781 navigation property of type reference that was included into the instances at the current
782 hop on behalf of some other navigation path.
783 `nav-hop` then identifies the instance or instances referenced by `ref-name`.
 - 784 – If `assoc-class-name` is specified in `nav-hop`, `assoc-class-name` shall be the name
785 of an association class that references one of the classes (including subclasses) of the
786 instances at the current hop.
787 `nav-hop` then identifies the instance or instances referenced by `ref-name` in `filtered-`
788 `ref`.
- 789 • `nav-filter`, when specified at a hop, filters the set of instances at that hop to be only
790 instances of class `filter-class-name` (including instances of its subclasses). Note that such
791 filtering can be used with both `ref-name` and `assoc-class-name`.

- 812 • Navigation paths that end with an association class name identify these association instances
- 813 For each navigation path in the `$expand` and `$refer` query parameters, a navigation property is
814 included in the retrieved instance representations, unless a reference property (typed or qualified) with
815 that name already exists. If two or more navigation paths can be merged, only one navigation property is
816 included that has the merged name and value, as described in the following paragraphs.
- 817 For the purpose of merging of navigation paths, the set of navigation paths in the `$expand` and `$refer`
818 query parameters is treated as one single combined set.
- 819 Two navigation paths can be merged if the first navigation path is a subset of the second navigation path,
820 and the first navigation path was used with `$expand`. Note that all navigation paths used in a particular
821 instance retrieval have the same starting point (the instance being retrieved).
- 822 The value of the merged navigation property is determined by identifying all elements (association
823 instances or references) in the value of the (expanded) property that would result from the first navigation
824 path alone, that are the starting points for the remainder of the second navigation path (that is, the
825 remaining string in the second navigation path after removing the portion that matches the first navigation
826 path), and by processing that remainder as a normal navigation path with the identified starting points.
827 Note that this can lead to both, expanding existing references, or including navigation properties.
- 828 The resulting merged property is considered to be included by `$expand`, for the purpose of applying the
829 merge rule repeatedly in cases where more than two navigation properties are merged. The repeated
830 merging of two navigation properties shall be performed in the order from the shortest to the longest
831 navigation path, regardless of the order in which they were specified in the `$expand` and `$refer` query
832 parameters.
- 833 The name of a navigation property is the navigation path string without any filter classes, or the subset
834 thereof that is a valid navigation path for the navigation property given the position of the navigation
835 property in the represented instance. See D.1 for examples on these names.
- 836 The values of navigation properties depend on whether `$expand` or `$refer` was used to include them;
837 for details see 6.5.3 and 6.5.9.

838 **EXPERIMENTAL**

839 **5.7 Discovering resources in a server**

- 840 This subclause provides an overview on how a client would go about discovering resources in a server,
841 using the CIM-RS protocol.
- 842 DMTF defines the use of SLP based discovery using the information in the *DMTF WBEM SLP Template*
843 ([DSP0206](#)). Clients can discover servers using this means (see clause 10). However, as with any WBEM
844 protocol, CIM-RS can be used without depending SLP, as long as the server is known by some means.
- 845 CIM-RS defines a well-known server entry point resource that may be used as a starting point for
846 discovery. Given a server URL, the client may retrieve the server entry point resource of the server using
847 an HTTP GET (see 7.12.2), using a resource identifier constructed using the well-known path component
848 of the server entry point resource (see 7.12).
- 849 The server entry point resource (and the listener entry point resource) are the only resources with a well-
850 known path component in their resource identifiers. Any other resource identifiers in CIM-RS are opaque
851 to clients.
- 852 Given a starting resource, the functionality of CIM-RS enables a client to navigate to all related resources.
853 The DMTF standard way of discovering implemented models and their entry points is described in the

854 *DMTF Profile Registration Profile* ([DSP1033](#)). The server entry point provides sufficient information for a
855 client to then utilize that standard.

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

861 **5.8 REST architectural style supported by CIM-RS**

862 CIM-RS follows most of the principles and constraints of the REST architectural style described by Roy
863 Fielding in chapter 5 of *Architectural Styles and the Design of Network-based Software Architectures* and
864 in *REST APIs must be hypertext driven*. Any deviations from these principles and constraints are
865 described in this subclause.

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

- 867 • **Client-Server:** The participants in CIM-RS have a client-server relationship between a WBEM
868 client and a WBEM server. For indication delivery, there is another client-server relationship in
869 the opposite direction: The WBEM server acting as a client operates against a WBEM listener
870 acting as a server. This constraint is fully satisfied.
- 871 • **Stateless:** Interactions in CIM-RS are self-describing and stateless in that the WBEM server or
872 the WBEM listener do not maintain any session state. This constraint is fully satisfied.

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

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

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

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

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

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

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

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

905 In CIM-RS, resources are addressed through resource identifiers that are URIs. The REST architectural
906 style recommends that all addressing information for a resource is in the resource identifier (and not, for
907 example, in the HTTP header). In addition, it recommends that resource identifiers are opaque to clients
908 and clients should not be required to understand the structure of resource identifiers or be required to
909 assemble any resource identifiers. CIM-RS follows the recommendations that all addressing information
910 for a resource is in the resource identifier and on opaqueness and non-assembly of the resource
911 identifier.

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

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

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

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

928 6 Resource identifiers

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

933 Usages of the resource identifier URI in the HTTP header are defined in [RFC2616](#) and [RFC2818](#). In the
934 protocol payload, resource identifiers are values of type URI (see Table 5), using the format defined in
935 6.1.

936 6.1 CIM-RS resource identifier format

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

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

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

940 Where:

- 941 • `authority` is defined in [RFC3986](#) and shall in addition conform to the definitions in 6.4
- 942 • `path-absolute` is defined in [RFC3986](#)

943 • `query` is defined in [RFC3986](#) and shall in addition conform to the definitions in 6.5

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

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

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

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

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

956 6.2 Opaqueness

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

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

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

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

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

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

977 6.3 Percent-encoding

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

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

- 981 • *Unreserved* characters (that is, the characters in ABNF rule `unreserved` defined in [RFC3986](#))
982 should not be percent-encoded. If they are percent-encoded, consumers of the resource
983 identifier shall tolerate that.
- 984 • The percent-encoding of *reserved* characters (that is, the characters in ABNF rule `reserved`
985 defined in [RFC3986](#)) depends on the specific query parameter and whether a character is
986 considered delimiter or data in that query parameter, or sometimes even within portions of the
987 query parameter.
- 988 Reserved characters that are considered delimiters shall not be percent-encoded.
- 989 Reserved characters that are considered data shall be percent-encoded.
- 990 The definitions of the query parameters in 6.5 defines which of the reserved characters are
991 considered delimiters or data, for purposes of percent-encoding.
- 992 • Any other characters (that is, outside of the ABNF rules `reserved` and `unreserved` defined in
993 [RFC3986](#)) shall be percent-encoded.
- 994 Consumers of resource identifiers shall support any percent-encoding within the resource identifier that is
995 permissible according to the rules in this subclause.
- 996 [RFC3986](#) defines percent-encoding on the basis of data octets, but it does not define how characters are
997 encoded as data octets. Because element names, namespace names, and key values may contain UCS
998 characters outside of the US-ASCII character set, this document defines the percent-encoding to be used
999 in resource identifiers as follows.
- 1000 Any UCS character that is being percent-encoded in resource identifiers shall be processed by first
1001 normalizing the UCS character using Normalization Form C (defined in [The Unicode Standard, Annex](#)
1002 [#15](#)), then encoding it to data octets using UTF-8, and finally percent-encoding the resulting data octets
1003 as defined in section 2.1 of [RFC3986](#). The requirement to use a specific Unicode normalization form and
1004 a specific Unicode encoding (that is, UTF-8) ensures that the resulting string can be compared octet-wise
1005 without having to apply UCS character semantics.
- 1006 If values with CIM datatypes need to be represented in resource identifiers, the datatype-specific string
1007 representations defined in [DSP0004](#) should be used.
- 1008 The following examples use the minimally needed percent-encodings:
- 1009 • The namespace name "root/cimv2" becomes "root%2Fcimv2" in a resource identifier, because
1010 the slash character (/) is a reserved character in resource identifiers and we assume that the
1011 usage of the namespace name has defined that an occurrence of a slash in a namespace name
1012 is considered data.
- 1013 • The class name "ACME_LogicalDevice" remains unchanged in a resource identifier, because it
1014 contains only unreserved characters.
- 1015 • The (German) key property value "ÄnderungsRate" becomes "%C3%84%0AnderungsRate" in a
1016 resource identifier, because C3 84 0A are the data octets of the UTF-8 encoding of the UCS
1017 character U+00C4, which represents "Ä" (A umlaut) in normalized form. Note that usage of the
1018 UCS character sequence U+0061 U+0308 which also represents "Ä" (using the base character
1019 "A" and the combining diacritical mark ¨) is not permitted due to the requirement to use
1020 Normalization Form C.
- 1021 • The string typed value "a \"brown\" bag\n" (represented using backslash escape sequences as
1022 defined for string literals in MOF) becomes "a%20%22brown%22%20bag%0A" in a resource
1023 identifier, because the characters blank (U+0020), newline (U+000A), and double quote
1024 (U+0022) are not allowed in resource identifiers and therefore need to be percent-encoded.

- 1025 • The sint8 typed value -42 becomes the string "-42" in a resource identifier, because that is the
1026 string representation of an sint8 typed value defined in [DSP0004](#), and because "-" is an
1027 unreserved character.

1028 6.4 Authority component

1029 Wbem clients, servers, and listeners shall adhere to the following additional rules regarding the value of
1030 ABNF rule `authority` defined in 6.1:

- 1031 • The `userinfo` component within `authority` shall not be specified because of security issues
1032 with specifying an unencrypted password
- 1033 • The `host` component within `authority` shall be the IP (V4 or V6) address of the server, or a
1034 DNS-resolvable host name for that IP address (including "localhost")
- 1035 • If the `port` component within `authority` is not specified, the port number shall default to the
1036 standard port numbers for HTTP and HTTPS:
 - 1037 – port number 80 when using HTTP
 - 1038 – port number 443 when using HTTPS

1039 If the authority component is omitted in values of type URI (see Table 5) in a request or response
1040 payload, it shall default to the authority used for that operation (that is, to the value of the Host request-
1041 header).

1042 6.5 Query parameters

1043 This subclause defines the query component of resource identifiers, and applies in addition to the
1044 definition in [RFC3986](#), section 3.4.

1045 The format of the query component is defined by the following ABNF rule:

```
1046 query = query-parameter *( "&" query-parameter )
```

1047 Where:

- 1048 • `query-parameter` is a query parameter as defined in the subclauses of this subclause
- 1049 • The reserved character "&" in the literals of this ABNF rule shall be considered a delimiter for
1050 purposes of percent-encoding (see 6.3)

1051 Example:

- 1052 • `/cimrs/networkports?$filter=Name='eth0'&$properties=Name,Description`

1053 This resource identifier specifies the query parameters `$filter` with a value of `Name='eth0'`
1054 and `$properties` with a value of `Name,Description`

- 1055 • `/cimrs/networkports?$filter=Description='a%26b'`

1056 This resource identifier specifies the query parameter `$filter` with a value of
1057 `Description='a&b'`, percent-encoding the ampersand character since it is considered a
1058 delimiter in the query parameter

1059 Query parameters of resource identifiers (that is, both name and value) are case sensitive, as defined in
1060 [RFC3986](#), section 6.2.2.1, unless defined otherwise in this subclause. The query parameters defined in
1061 the subclauses of this subclause define in some cases that the values of query parameters are to be
1062 treated case insensitively. In such cases, two resource identifiers that differ only in the lexical case of
1063 query parameters address the same resource, even though the resource identifiers do not match
1064 according to the rules defined in [RFC3986](#). It is recommended that producers of resource identifiers

1065 preserve the lexical case in such case insensitive cases, in order to optimize caching based on resource
 1066 identifiers. For example, if a property is named "ErrorRate", its use in the \$properties query parameter
 1067 should be "\$properties=ErrorRate", preserving its lexical case.

1068 Query parameters whose syntax supports the specification of comma-separated lists of items may be
 1069 repeated; the effective list of items is the concatenation of all those lists. Any other query parameters shall
 1070 not be repeated (unless specified otherwise in the description of the query parameter); if such query
 1071 parameters are repeated in a resource identifier, the consumer of that resource identifier shall fail the
 1072 operation with HTTP status code 400 "Bad Request". The description of each query parameter will detail
 1073 whether it permits repetition.

1074 NOTE: [RFC3986](#) does not detail how the query ABNF rule is broken into query parameters, and thus does not
 1075 address the topic of query parameter repetition.

1076 The order and repetition of query parameters specified in resource identifiers does not matter for
 1077 purposes of identifying the resource and for the semantic of the query parameters. As a consequence,
 1078 resource identifiers need to be normalized before a simple string comparison can be used to determine
 1079 resource identity.

1080 Some query parameters are constrained to be specified only on certain resource identifiers, as defined in
 1081 the subclauses of this subclause. WBEM servers and listeners shall reject operations against resource
 1082 identifiers that do not conform to these constraints.

1083 This subclause defines the query-parameter rule by using ABNF incremental alternatives (that is, the
 1084 =/ construct), based on the initially empty rule:

1085 `query-parameter = "" ; initially empty`

1086 Table 1 lists the query parameters that shall be supported, subject to the usage constraints defined in this
 1087 document:

1088 **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 (EXPERIMENTAL)	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 (EXPERIMENTAL)	include references to target instances	see 6.5.9

1089 Additional implementation-defined query parameters are not permitted in CIM-RS. Note that servers (and
 1090 listeners) can use the path component of a resource identifier to include any implementation-defined
 1091 information (as long as it is opaque to the receivers).

1092 In order to prepare for query parameters to be added in future versions of this document, clients, servers
 1093 and listeners shall tolerate and ignore any query parameters not listed in Table 1. As a result, two

1094 resource identifiers that differ only in the presence of a query parameter not listed in Table 1 address the
1095 same resource.

1096 **6.5.1 \$class (specify class name)**

1097 The `$class` query parameter is used to specify a class name for the HTTP PUT method on instance
1098 enumeration resources (see 7.9.1) or the HTTP POST method on instance creation resources (see
1099 7.5.1).

1100 The format of this query parameter is defined by the following ABNF:

```
1101 query-parameter =/ class-query-parm
1102
1103 class-query-parm = "$class=" class-name
```

1104 Where:

- 1105 • The reserved characters "\$" and "=" in the literals of these ABNF rules shall be considered
1106 delimiters for purposes of percent-encoding (see 6.3)
- 1107 • `class-name` is the name of the class (including schema prefix). Note that CIM class names do
1108 not contain reserved characters (see 6.3 and [DSP0004](#))

1109 The `$class` query parameter shall not be repeated in a resource identifier.

1110 Examples:

```
1111 $class=ACME_ComputerSystem
1112
1113     specifies class name ACME_Computersystem
```

1113 **6.5.2 \$continueonerror (continue on errors within paged retrieval)**

1114 The `$continueonerror` query parameter specifies whether or not the server continues paged retrieval
1115 sequences in case of errors (instead of closing them). For details about paged retrieval, see 7.3.8.

1116 The format of this query parameter is defined by the following ABNF:

```
1117 query-parameter =/ continueonerror-query-parm
1118
1119 continueonerror-query-parm = "$continueonerror" [ "=" ( "true" / "false" ) ]
```

1120 Where:

- 1121 • The reserved characters "\$" and "=" in the literals of these ABNF rules shall be considered
1122 delimiters for purposes of percent-encoding (see 6.3)

1123 Note that the values "true" and "false" are treated case sensitively, as defined in 6.3

1124 The `$continueonerror` query parameter shall not be repeated in a resource identifier.

1125 Omitting the `$continueonerror` query parameter or specifying it with a value of "false" shall cause the
1126 server to close paged retrieval sequences in case of errors.

1127 Specifying the `$continueonerror` query parameter without a value or with a value of "true" shall cause
1128 the server to continue paged retrieval sequences in case of errors.

1129 Examples:

1130 (not specified)
 1131 `$continueonerror=false`

1132 The server closes paged retrieval sequences in case of errors

1133 `$continueonerror`
 1134 `$continueonerror=true`

1135 The server continues paged retrieval sequences in case of errors

1136 6.5.3 \$expand (include target instances, EXPERIMENTAL)

1137 EXPERIMENTAL

1138 The `$expand` query parameter may be used on operations that retrieve instances or instance collections
 1139 and specifies a list of navigation paths. For details on navigation paths and the resulting navigation
 1140 properties, see 5.6.

1141 The value of navigation properties included as a result of using the `$expand` query parameter shall be an
 1142 instance collection whose members are the target instances identified by the navigation path. That
 1143 instance collection shall be represented as an InstanceCollection payload element (see 7.8.1) and shall
 1144 be subject to paged retrieval (see 7.3.8).

1145 The value of existing references expanded as a result of using the `$expand` query parameter depends on
 1146 the navigation path, as follows. Note that the navigation path may contain more than one hop:

- 1147 • if each hop on the navigation path is a scalar reference (typed or qualified), the value of the
 1148 expanded reference shall be the target instance identified by the navigation path. That instance
 1149 shall be represented as an Instance payload element (see 7.6.1).
- 1150 • otherwise, the value of the expanded reference shall be an instance collection whose members
 1151 are the target instances identified by the navigation path. That instance collection shall be
 1152 represented as an InstanceCollection payload element (see 7.8.1) and shall be subject to paged
 1153 retrieval (see 7.3.8).

1154 The format of the `$expand` query parameter is defined by the following ABNF:

```
1155 query-parameter =/ expand-query-parm
1156
1157 expand-query-parm = "$expand=" [ expand-list ]
1158
1159 expand-list = nav-path *( "," nav-path )
```

1160 Where:

- 1161 • The reserved characters "\$", "=", and "," in the literals of these ABNF rules shall be considered
 1162 delimiters for purposes of percent-encoding (see 6.3)
- 1163 • `nav-path` is a navigation path identifying the target instances, as defined in 5.6; any reserved
 1164 characters in the navigation path (that is, "[" and "]") shall be considered delimiters for purposes
 1165 of percent-encoding (see 6.3). Note that the character "." in the navigation path is an
 1166 unreserved character.

1167 The `$expand` query parameter may be repeated in a resource identifier, see 6.5. If repeated, the
 1168 effective expand list shall be the combined expand list of all occurrences of the `$expand` query
 1169 parameter.

1170 Duplicate or invalid navigation path strings in the set of all navigation paths specified for the `$expand` or
 1171 `$refer` query parameters shall cause the operation to fail with HTTP status code 400 "Bad Request".

1172 Examples:

1173 (not specified)

1174 `$expand=`

1175 no navigation paths have been specified; no navigation properties will be included and no
 1176 expansion of reference properties will take place

1177 `$expand=ACME_SystemDevice.PartComponent`

1178 include a navigation property named "ACME_SystemDevice.PartComponent" in each retrieved
 1179 instance (assuming it is valid for the retrieved instance)

1180 `$expand=Volumes`

1181 expand the reference-qualified property array named "Volumes", to an instance collection of the
 1182 referenced instances.

1183 For more examples, see D.1.

1184 EXPERIMENTAL

1185 6.5.4 `$filter` (filter instances in result)

1186 The `$filter` query parameter acts as a restricting filter on the set of instances included in an instance
 1187 collection.

1188 In this version of CIM-RS, the only query language supported for the `$filter` query parameter is the
 1189 DMTF *Filter Query Language* (FQL) defined in [DSP0212](#).

1190 The format of this query parameter is defined by the following ABNF:

```
1191 query-parameter =/ filter-query-parm
1192
1193 filter-query-parm = "$filter=" [ filter-query ]
```

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 • `filter-query` is a filter query string that shall conform to the format of an FQL query string; if
 1198 it evaluates to true for an instance then the instance is included, otherwise, it is not included.

1199 Any reserved characters that occur in literals of the FQL query string shall be considered data
 1200 for purposes of percent-encoding.

1201 Any reserved characters that occur elsewhere in the FQL query string shall be considered
 1202 delimiters for purposes of percent-encoding (see 6.3).

1203 The `$filter` query parameter may be repeated in a resource identifier, see 6.5. Multiple occurrences of
 1204 the `$filter` query parameter shall be combined by using logical AND on the filter query of each single
 1205 parameter value.

1206 The `$filter` query parameter may be specified only in resource identifiers of instance collection
 1207 resources.

1208 Navigation properties cannot be specified in the FQL query string. If navigation properties are specified in
1209 the FQL query string, the server shall fail the operation with HTTP status code 400 "Bad Request". This is
1210 motivated by the fact that FQL is a query language that remains local with the set of instances and by the
1211 desire to allow servers that internally use generic operations to pass the (decoded) FQL query string on
1212 without further processing it.

1213 Omitting the `$filter` query parameter shall result in no additional restrictive filtering of instances in the
1214 instance collection.

1215 A `$filter` query parameter that is specified with no value shall result in including no instances from the
1216 instance collection.

1217 Examples:

1218 (not specified)

1219 no additional restrictive instance filtering takes place

1220 `$filter=`

1221 includes no instances

1222 `$filter=Type='LAN'%20AND%20ErrorRate%3E0`

1223 specifies the FQL query string "`Type='LAN' AND ErrorRate>0`" and causes only instances
1224 with properties `Type = "LAN"` and `ErrorRate > 0` to be included.

1225 The reserved characters "=" and single quote (') in the FQL query string are not percent-
1226 encoded because they do not occur in literals of the FQL query string and are therefore
1227 considered delimiters.

1228 The blank and ">" characters are not allowed in resource identifiers and are therefore percent-
1229 encoded.

1230 `$filter=Description='a%2Cb%3D0'`

1231 specifies the FQL query string "`Description='a,b=0'`" and causes only instances with
1232 property `Description = "a,b=0"` to be included.

1233 The first occurrence of the reserved character "=" in the FQL query string (right after
1234 Description) is not percent-encoded because it does not occur in literals of the FQL query string
1235 and is therefore considered a delimiter.

1236 The second occurrence of the reserved character "=" and the reserved character "," in the FQL
1237 query string (in the Description value) are percent-encoded because they occur in a literal of the
1238 FQL query string and are therefore considered data.

1239 **6.5.5 \$max (limit number of collection members in result)**

1240 The `$max` query parameter limits the number of members in any retrieved collections to the specified
1241 number.

1242 If there are members in excess of that maximum number, the server shall return the collection in paged
1243 mode. Note that a server may choose to return the collection in paged mode also when the specified
1244 maximum number of members is not exceeded. For details on paging of collections, see 7.3.8.

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

```

1246 query-parameter =/ max-query-parm
1247
1248 max-query-parm = "$max=" max-members
1249
1250 max-members = nonNegativeDecimalInteger

```

1251 Where:

- 1252 • The reserved characters "\$" and "=" in the literals of these ABNF rules shall be considered
- 1253 delimiters for purposes of percent-encoding (see 6.3)
- 1254 • `max-members` specifies the maximum number of collection members.

1255 The `$max` query parameter shall not be repeated in a resource identifier.

1256 Omitting the `$max` query parameter indicates that there is no maximum number specified.

1257 Specifying the `$max` query parameter with a value of 0 indicates that a collection with no members shall
1258 be returned.

1259 Note that a server may choose to use paging also when the no maximum is specified.

1260 Examples:

1261 (not specified)

1262 no maximum is specified for the number of members in the collection result.

1263 `$max=0`

1264 number of members in the collection result is limited to no more than 0 (that is, the collection is
1265 empty).

1266 `$max=10`

1267 number of members in the collection result is limited to no more than 10.

1268 6.5.6 \$methods (subset method links in result)

1269 The `$methods` query parameter subsets the method invocation links any instances or instance
1270 collections to only those for the specified set of method names.

1271 The format of this query parameter is defined by the following ABNF:

```

1272 query-parameter =/ methods-query-parm
1273
1274 methods-query-parm = "$methods=" [ method-list ]
1275
1276 method-list = method-spec *( "," method-spec )
1277
1278 method-spec = [ nav-path "." ] method-name

```

1279 Where:

- 1280 • The reserved characters "\$", "=", and "," in the literals of these ABNF rules shall be considered
- 1281 delimiters for purposes of percent-encoding (see 6.3). Note that the character "." used in the in
- 1282 the literals of these ABNF rules is an unreserved character.
- 1283 • `method-name` is the name of a method (without parenthesis or any method parameters)

1284 EXPERIMENTAL

- 1285 • `nav-path` is the navigation path to the instances whose method invocation links are to be
1286 subsetted. `nav-path` and the concept of a navigation path is described in 5.6. Any reserved
1287 characters in the navigation path (that is, "[" and "]") shall be considered delimiters for purposes
1288 of percent-encoding (see 6.3). Note that the character "." in the navigation path is an
1289 unreserved character.

1290 EXPERIMENTAL

1291 The `$methods` query parameter may be repeated in a resource identifier, see 6.5. If repeated, the
1292 effective method list shall be the combined method list of all occurrences of the `$methods` query
1293 parameter.

1294 Omitting the `$methods` query parameter shall result in not excluding any method invocation links.

1295 A `$methods` query parameter that is specified with no value shall result in including no method invocation
1296 links in the instances, instance collections or instances in the instance collections.

1297 This query parameter may be specified only in resource identifiers of instance resources or instance
1298 collection resources. If specified in resource identifiers of instance collection resources, it applies to the
1299 instance collection itself and to all instances in the collection.

1300 EXPERIMENTAL

1301 Any navigation path used to identify method invocation links shall also be specified in the `$expand` query
1302 parameter. This ensures that the instances of such links are part of the retrieved instance
1303 representations. If this condition is not met, the consumer shall fail the operation with HTTP status code
1304 400 "Bad Request".

1305 EXPERIMENTAL

1306 Duplicate and invalid method names shall be ignored. Invalid method names are names of methods that
1307 are not exposed by the creation class of an instance.

1308 Examples:

1309 (not specified)

1310 no method invocation links are excluded

1311 `$methods=`

1312 no method invocation links are included

1313 `$methods=Start, Stop`

1314 only the method invocation links for methods "Start" and "Stop" are included

1315 6.5.7 \$pagingtimeout (specify inactivity timeout for paged retrieval)

1316 The `$pagingtimeout` query parameter specifies a duration after which a server may close a sequence
1317 of paged retrievals of subset collections if there is no retrieval activity on that sequence. This duration is
1318 referred to as *paging timeout*. For details, see 7.3.8.

1319 The format of this query parameter is defined by the following ABNF:

```

1320 query-parameter =/ pagingtimeout-query-param
1321
1322 pagingtimeout-query-param = "$pagingtimeout=" duration
1323
1324 duration = nonNegativeDecimalInteger

```

1325 Where:

- 1326 • The reserved characters "\$" and "=" in the literals of these ABNF rules shall be considered
1327 delimiters for purposes of percent-encoding (see 6.3)
- 1328 • `duration` is the duration of the paging timeout in seconds. A value of 0 specifies that there is
1329 no paging timeout (that is, an infinite paging timeout)

1330 The `$pagingtimeout` query parameter shall not be repeated in a resource identifier.

1331 Omitting the `$pagingtimeout` query parameter shall result in using the default paging timeout of the
1332 server (see 7.12).

1333 The allowable values for the paging timeout clients may specify with the `$pagingtimeout` query
1334 parameter can be discovered by clients through the "minimumpagingtimeout" and
1335 "maximumpagingtimeout" attributes of the server entry point resource (see 7.12).

1336 Examples:

1337 (not specified)

1338 default paging timeout of the server is used

1339 `$pagingtimeout=0`

1340 no paging timeout is used (infinite paging timeout)

1341 `$pagingtimeout=30`

1342 a paging timeout of 30 seconds is used

1343 6.5.8 \$properties (subset properties in result)

1344 The `$properties` query parameter subsets the properties in any retrieved instance representations to
1345 only the specified set of properties. This is semantically equivalent to acting on a different resource that is
1346 a subset of the full resource.

1347 The format of this query parameter is defined by the following ABNF:

```

1348 query-parameter =/ properties-query-param
1349
1350 properties-query-param = "$properties=" [ property-list ]
1351
1352 property-list = property-spec *( "," property-spec )
1353
1354 property-spec = [ nav-path "." ] property-name

```

1355 Where:

- 1356 • The reserved characters "\$", "=", and "," in the literals of these ABNF rules shall be considered
1357 delimiters for purposes of percent-encoding (see 6.3). Note that the character "." used in the in
1358 the literals of these ABNF rules is an unreserved character.

- 1359
- `property-name` is the name of a property in the instances
-

1360 **EXPERIMENTAL**

- 1361
- `nav-path` is the navigation path to the instances whose properties are to be subsetted. `nav-path` and the concept of a navigation path is described in 5.6. Any reserved characters in the navigation path (that is, "[" and "]") shall be considered delimiters for purposes of percent-encoding (see 6.3). Note that the character "." in the navigation path is an unreserved character.

1365 **EXPERIMENTAL**

1366 The `$properties` query parameter may be repeated in a resource identifier, see 6.5. If repeated, the
 1367 effective property list shall be the combined property list of all occurrences of the `$properties` query
 1368 parameter.

1369 Omitting the `$properties` query parameter shall result in not excluding any properties.

1370 A `$properties` query parameter that is specified with no value shall result in including no properties in
 1371 the retrieved instance representations.

1372 The order of property names specified in the query parameter is not relevant for the order of properties in
 1373 the retrieved instance representations.

1374 This query parameter may be specified only in resource identifiers of instance resources or instance
 1375 collection resources. If specified in resource identifiers of instance collection resources, it applies to all
 1376 instances in the collection.

1377 Any navigation path used to identify properties shall also be specified in the `$expand` query parameter.
 1378 This ensures that the instances of such properties are part of the retrieved instance representations. If
 1379 this condition is not met, the consumer shall fail the operation with HTTP status code 400 "Bad Request".

1380 Duplicate and invalid property names shall be ignored. Invalid property names are names of properties
 1381 that are not exposed by the creation class of an instance.

1382 Examples:

1383 (not specified)

1384 no properties are excluded

1385 `$properties=`

1386 no properties are included

1387 `$properties=Name, Type`

1388 only the properties "Name" and "Type" are included

1389 **6.5.9 \$refer (include references to target instances, EXPERIMENTAL)**

1390 **EXPERIMENTAL**

1391 The `$refer` query parameter may be used on operations that retrieve instances or instance collections
 1392 and specifies a list of navigation paths. For details on navigation paths and the resulting navigation
 1393 properties, see 5.6.

1394 The value of navigation properties included as a result of using the `$refer` query parameter shall be a
 1395 reference collection whose members are references to the target instances identified by the navigation
 1396 path. That reference collection shall be represented as a ReferenceCollection payload element (see
 1397 7.7.1) and shall be subject to paged retrieval (see 7.3.8).

1398 Navigation paths that refer to existing references (qualified or typed, scalar or array) can be used to
 1399 subset these references in the retrieved instance representations by specifying `filter-class-name` in
 1400 the navigation path (see 5.6).

1401 The format of the `$refer` query parameter is defined by the following ABNF:

```
1402 query-parameter =/ refer-query-parm
1403
1404 refer-query-parm = "$refer=" [ refer-list ]
1405
1406 refer-list = nav-path *( "," nav-path )
```

1407 Where:

- 1408 • The reserved characters "\$", "=", and ",", in the literals of these ABNF rules shall be considered
 1409 delimiters for purposes of percent-encoding (see 6.3).
- 1410 • `nav-path` is a navigation path identifying target instances, as defined in 5.6. Any reserved
 1411 characters in the navigation path (that is, "[" and "]") shall be considered delimiters for purposes
 1412 of percent-encoding (see 6.3). Note that the character "." in the navigation path is an
 1413 unreserved character.

1414 The `$refer` query parameter may be repeated in a resource identifier, see 6.5. If repeated, the effective
 1415 refer list shall be the combined refer list of all occurrences of the `$refer` query parameter.

1416 Duplicate or invalid navigation path strings in the set of all navigation paths specified for the `$expand` or
 1417 `$refer` query parameters shall cause the operation to fail with HTTP status code 400 "Bad Request".

1418 Examples:

1419 (not specified)
 1420 `$refer=`

1421 No navigation paths have been specified; no navigation properties will be included

1422 `$refer=ACME_SystemDevice.PartComponent,ACME_HostedService.Service`

1423 include navigation properties named "ACME_SystemDevice.PartComponent" and
 1424 "ACME_HostedService.Service" in each retrieved instance (assuming both are valid for the
 1425 retrieved instance)

1426 For more examples, see D.1.

1427 EXPERIMENTAL

1428 6.6 Resource identifiers of entry point resources

1429 The server and listener entry point resources are the only resources in the CIM-RS protocol that have
 1430 well-known resource identifiers.

1431 The resource identifier of the server entry point resource of a server shall have the path component
 1432 defined by the following ABNF rule:

1433 `server-entry-point-path = "/cimrs" ["/"]`

1434 The resource identifier of the listener entry point resource of a listener shall have the path component
1435 defined by the following ABNF rule:

1436 `listener-entry-point-path = "/cimrs" ["/"]`

1437 Examples:

1438 `/cimrs`

1439 `//acme.com/cimrs/`

1440 7 Resources, operations and payload elements

1441 This clause defines the types of resources used in the CIM-RS protocol, the operations on these
1442 resources, and the payload elements used in the protocol payload when performing these operations.

1443 7.1 Overview

1444 Table 2 shows an overview of all types of resources used in the CIM-RS protocol. A resource in the CIM-
1445 RS protocol is anything that can be the target of an HTTP method.

1446 **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 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 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
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 destination resource	A resource within a listener that can be used to deliver indications
Listener entry point resource	The entry point resource of a listener, representing capabilities of the listener

1447 A combination of a particular HTTP method on a particular type of resource is termed an "operation" in
1448 this document. For ease of reference by other documents, these operations have names. However, the
1449 names of the operations do not show up in the protocol.

1450 Table 3 shows all operations used in the CIM-RS protocol, identified by their HTTP method and target
1451 resource type.

1452

Table 3 – CIM-RS operations

HTTP Method	Target Resource Type	Description
DELETE	Instance resource	see 7.6.2
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

1453 Most of the operations used in the CIM-RS protocol have protocol payload data either in the request
 1454 message, or in the response message, or both. These payload elements often correspond directly to
 1455 resources, but not always. This document defines these payload elements in a normative but abstract
 1456 way. CIM-RS payload representation specifications define how each of these payload elements is
 1457 represented, for details see clause 9. The payload elements have a name for ease of referencing
 1458 between documents, as shown in the first column of Table 4.

1459 Table 4 shows all payload elements used in the CIM-RS protocol.

1460

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
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

1461

1462 **7.2 Description conventions**

1463 **7.2.1 Datatypes used in payload element definitions**

1464 This subclause defines the datatypes used in the definition of the attributes of payload elements. In order
 1465 to distinguish these kinds of datatypes from CIM datatypes, they are termed "payload datatypes". Payload
 1466 datatypes are used as a description mechanism for this document and for any payload representation
 1467 specifications.

1468 The representation of values of payload datatypes is defined in payload representation specifications; for
 1469 details see clause 9.

1470 **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: <table border="1" data-bbox="402 829 1432 1073"> <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 method (without any parenthesis or method parameters)</td> </tr> <tr> <td>class</td> <td>String</td> <td>name of the implemented class exposing the method</td> </tr> <tr> <td>uri</td> <td>URI</td> <td>resource identifier of the method invocation resource (see 7.10)</td> </tr> </tbody> </table>	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)
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)											
ElementValue	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: <table border="1" data-bbox="402 1176 1432 1501"> <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												

1471 The CIM datatype specified in the "type" child element of the ElementValue type allows infrastructure
 1472 components to represent element values in programming environments using strong types for the CIM
 1473 datatypes. This is expected to be used for WBEM client implementations as model-neutral client libraries.

1474 Representation of the "type" child element of the ElementValue payload datatype is optional for payload
 1475 representations. If a payload representation supports representation of the "type" child element, it shall be
 1476 present; otherwise, it shall be omitted. Note that this decision is made by the definition of a payload
 1477 representation, and not by an implementation of CIM-RS.

1478 7.2.2 Requirement levels used in payload element definitions

1479 This subclause defines the meaning of requirement levels used in the definition of the attributes of
1480 payload elements.

1481	Mandatory	The attribute shall be included in the payload element.
1482	Conditional	The attribute shall be included in the payload element if the condition is
1483		met. If the condition is not met, the attribute may be included in the
1484		payload element at the discretion of the implementation.
1485	ConditionalExclusive	The attribute shall be included in the payload element if the condition is
1486		met. If the condition is not met, the attribute shall not be included in the
1487		payload element.
1488	Optional	The attribute may be included in the payload element at the discretion of
1489		the implementation.

1490 7.2.3 Requirement levels used in operation definitions

1491 This subclause defines the meaning of requirement levels used in the descriptions of operations:

1492	Mandatory	The operation shall be implemented. It is not expected that the
1493		implementation of the operation is specific to a class or model.
1494	Mandatory (class specific)	The implementation of the operation is specific to a class or model.
1495		General infrastructure support for the operation (that is, functionality not
1496		specific to a class or model) shall be implemented; the requirements for
1497		implementing the operation for specific classes are defined elsewhere
1498		(for example, in management profiles)

1499 7.2.4 CIM-RS operation description format

1500 The definition of operations in the following subclauses uses the following description fields:

1501	Name:	The name of the operation.
1502	Purpose:	A brief description of the purpose of the operation.
1503	HTTP method:	The name of the HTTP method used to perform the operation (for
1504		example, GET, PUT, POST, DELETE).
1505	Target resource:	The resource that is identified as the target of the HTTP method, by
1506		means of the Request-URI field (see RFC2616) and Host header field.
1507	Query parameters:	The names of any query parameters that may be specified in the
1508		resource identifier. Other query parameters shall not be specified by the
1509		requester. If other query parameters are specified by the requester, they
1510		shall be ignored by the responder, in order to provide for future
1511		extensibility.
1512	Request headers:	The names of any header fields that may be specified in the request
1513		message. Other request headers shall not be specified by the requester.
1514		If other query request headers are specified by the requester, they shall
1515		be ignored by the responder, in order to provide for future extensibility.
1516	Request payload:	The name of the payload element that shall be used in the entity body of
1517		the request message. "None" means the entity body shall be empty.

1518	Response headers:	The names of any header fields that may be specified in the response message, separately for the success and failure case. Other response headers shall not be specified by the responder. If other query request headers are specified by the responder, they shall be ignored by the requester, in order to provide for future extensibility.
1519		
1520		
1521		
1522		
1523	Response payload:	The name of the payload element that shall be used in the entity body of the response message, separately for the success and failure case. "None" means the entity body shall be empty.
1524		
1525		
1526	Requirement:	The requirement level to implement the operation, as defined in 7.2.3.
1527	Description:	A normative definition of the behavior of the operation, in addition to the normative definitions stated in the previous description fields.
1528		
1529	Example HTTP conversation:	An example HTTP request and HTTP response. The examples use the CIM-RS payload representation in JSON defined in DSP0211 . In case of differences between these examples and DSP0211 , the latter wins.
1530		
1531		
1531		

1532 7.3 Common behaviors for all operations

1533 7.3.1 Content negotiation

1534 WBEM clients, servers, and listeners shall support server-driven content negotiation as defined in
1535 [RFC2616](#), based on the Accept request-header (defined in [RFC2616](#) and in 8.4.1), and the Content-Type
1536 response header field (defined in [RFC2616](#) and in 8.4.2).

1537 Requirements for the media types used in these header fields are defined in 9.1.

1538 The entry point resources of server and listener can be retrieved in order to discover the supported set of
1539 CIM-RS payload representations, as described in 7.12.2 and 7.13.2.

1540 7.3.2 Verifying the basis of resource modifications (EXPERIMENTAL)

1541 EXPERIMENTAL

1542 The HTTP PUT method on an instance resource (see 7.6.4) takes an instance with the new property
1543 values as input. The CIM-RS protocol provides for a means to verify for a server whether the current state
1544 of the resource is still the same as when the client retrieved the resource as a basis for the modifications.

1545 This may be achieved by using the value of the CIM Generation property (defined in ACME_Element) as
1546 an entity tag with the ETag and If-Match HTTP header fields, as described in 8.4.3 and 8.4.4.

1547 This ability is part of the optional entity tagging feature (see 7.4.1).

1548 EXPERIMENTAL

1549 7.3.3 Caching of responses

1550 Caching of responses from servers and listeners is described in [RFC2616](#). This document does not
1551 define any additional constraints or restrictions on caching.

1552 Note that any use of the HTTP GET method in the CIM-RS protocol is safe and idempotent, and that any
1553 use of the HTTP PUT method in the CIM-RS protocol is idempotent.

1554 Implementing the entity tagging feature (see 7.4.1) improves cache control.

1555 **7.3.4 Success and failure**

1556 Operations performed within the CIM-RS protocol shall either succeed or fail. There is no concept of
1557 "partial success".

1558 If an operation succeeds, it shall return its output data to the operation requester and shall not include any
1559 errors .

1560 If an operation fails, it shall return an error to the operation requester (see 7.3.6) and no output data.

1561 For example, if an instance collection retrieval operation were able to return some, but not all, instances
1562 successfully, then the operation fails without returning any instances.

1563 When using paged retrieval, each retrieval operation within a paged retrieval stream is considered a
1564 separate operation w.r.t. success and failure.

1565 **7.3.5 Errors**

1566 Errors at the CIM-RS protocol level are returned as HTTP status codes. The definition of HTTP status
1567 codes defined in [RFC2616](#) is the basis for each operation, and the operation descriptions in this
1568 document specify any additional constraints on the use of HTTP status codes.

1569 Extended error information is returned as an ErrorResponse payload element (see 7.3.6) in the entity
1570 body. For details about its usage, see the operation descriptions in clause 7.

1571 **7.3.6 ErrorResponse payload element**

1572 An ErrorResponse payload element represents the data used in an error response to any request.

1573 An ErrorResponse payload element shall have the following attributes:

1574 **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

1575

1576 **7.3.7 Consistency model**

1577 The operations of the CIM-RS protocol shall conform to the consistency model defined in [DSP0223](#).

1578 7.3.8 Paging of collections

1579 Client and servers shall support the *paging of collections* returned to clients as described in this
1580 subclause.

1581 An instance collection contains an order-preserving list of instance representations). When a
1582 representation of an instance collection is returned to a client, the server may choose to use paging for
1583 the instance collection, at the server's discretion.

1584 A reference collection contains an order-preserving list of references to instances. When a representation
1585 of a reference collection is returned to a client, the server may choose to use paging for the reference
1586 collection, at the server's discretion.

1587 If the server does not use paging for a collection, the "next" attribute of that collection shall be omitted.

1588 If the server uses paging for a collection, its "next" attribute shall reference a collection resource that
1589 contains the next subset of collection members. That next subset collection may again contain only a
1590 subset of the remaining members, and so forth. The last subset collection has no "next" attribute,
1591 indicating that it is the last one of the sequence of subset collections.

1592 The members in each subset collection form an order-preserving list, and appending the lists of these
1593 subset collections in the order of their "next" links shall reconstruct the original order of members in the
1594 entire collection. In other words, the order of members in a collection is maintained when paging is used
1595 to retrieve the collection.

1596 As a result, any InstanceCollection payload element (see 7.8.1) or ReferenceCollection payload element
1597 (see 7.7.1) is self-describing w.r.t. whether it contains the last (or possibly only) set of members, or other
1598 subsets are following; and the subdivision of the complete set of instances into subset collections always
1599 happens at a granularity of complete instances (that is, instances are never broken apart to be returned in
1600 separate subset collections).

1601 Instance collection and reference collection resources can be retrieved directly using the HTTP GET
1602 method.

1603 EXPERIMENTAL

1604 Instance collections and reference collections can also be part of instances (for example, when using the
1605 `$expand` or `$refer` query parameters, see 5.6). If an instance (being retrieved directly, or being part of
1606 an instance collection that is retrieved) contains instance collections or reference collections, these
1607 nested collections may also be paged, at arbitrary nesting depth. Servers may choose to page or not to
1608 page the collections in a result independently of each other.

1609 EXPERIMENTAL

1610 Clients and servers shall support paging of collections for the following operations:

1611

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

1612 Clients may use the $\$max$ query parameter (see 6.5.5) to limit the number of members in each returned
 1613 (subset) collection.

1614 Each returned (subset) collection shall contain any number of members between one and the maximum
 1615 specified with the $\$max$ query parameter (if specified). The number of members in a collection may
 1616 change between any two subset collections (belonging to the same or different entire collection, or
 1617 operation). As a result, the number of members in a collection is not a safe indicator for a client that there
 1618 are remaining members; only the presence of the "next" attribute is a safe indicator for that.

1619 Because the server decides about whether or not to page any collections, from a client's perspective the
 1620 resource identifier of a collection resource sometimes references the entire collection, and sometimes
 1621 only the first subset collection. As a result, the resources referenced by such resource identifiers
 1622 represent *possibly paged collections*.

1623 The resource identifiers of the set of subset collections representing a complete collection shall all be
 1624 distinct. Servers shall represent the state of retrieval progress within a sequence of subset collections in
 1625 the resource identifiers of the subset collections.

1626 Servers should implement ceasing of subset collection resources. If a server implements ceasing of
 1627 subset collection resources, successfully retrieved subsequent subset collections (that is, second to last)
 1628 shall cause the retrieved subset collection resource to cease existence, and subsequent requests to
 1629 retrieve that subset collection resource shall be rejected with HTTP status code 404 "Not Found".

1630 The first subset collection of a sequence shall not cease existence as a result of being successfully
 1631 retrieved, when the server implements ceasing of subset collection resources (however, it may cease
 1632 existence for other reasons, such as ceasing of the represented managed object). Separate retrieval
 1633 requests for the entire and first subset collection shall be treated independently by the server (regardless
 1634 of whether these requests come from the same or different clients, and regardless of whether a request is
 1635 a repetition of an earlier request). As a result, each successful retrieval request of the first subset
 1636 collection opens a new sequence of paged retrievals for the remaining subset collections.

1637 Clients and servers may support the continue on error feature (see 7.4.2). Clients that support the
 1638 continue on error feature may request continuation on error for paged retrievals by specifying the
 1639 $\$continueonerror$ query parameter (see 6.5.2). If a retrieval request results in an error, the client has
 1640 request continuation on error, and the server supports the continue on error feature, the server shall not
 1641 close the sequence of retrievals. Otherwise, the server shall close the sequence of retrievals, if a retrieval
 1642 request results in an error. For details on this behavior, see the description of "continuation on error" of
 1643 pulled enumerations in [DSP0223](#).

1644 Servers should close a sequence of paged retrievals after some time of inactivity on that sequence, even
 1645 if the client has not retrieved the sequence exhaustively. Clients may use the $\$pagingtimeout$ query
 1646 parameter (see 6.5.7) to specify the minimum duration the server is obliged to keep a sequence of paged
 1647 subset collections open after retrieval of a subset collection. If the $\$pagingtimeout$ query parameter is
 1648 not specified, the server default shall be used, which is indicated in the "defaultPagingTimeout" attribute
 1649 of the server entry point resource (see 7.12). For details on this behavior, see the description of
 1650 "operation timeout" of pulled enumerations in [DSP0223](#).

1651 The concept of paging collections as described in this subclause is consistent with pulled enumerations
1652 as defined in [DSP0223](#), so that it fits easily with servers that support the semantics of pulled
1653 enumerations in their implementation.

1654 Servers that support pulled enumerations in their implementation can achieve to be entirely stateless
1655 w.r.t. paging collections, by maintaining the entire state data of the paging progress in the enumeration
1656 context value, and by representing the enumeration context value in the resource identifiers of
1657 subsequent (second to last) subset collections. Binary data in an enumeration context value can for
1658 example be represented using a base64url encoding (see [RFC4648](#)), typically without any "=" padding
1659 characters at the end.

1660 For more details on pulled enumerations and the concept of enumeration context values, see [DSP0223](#).

1661 NOTE: The use of HTTP range requests as defined in [RFC2616](#) has been considered and dismissed, because the
1662 semantics of an ordered sequence of items that can be accessed by item number cannot be provided by
1663 implementations that support the opaque server-defined enumeration context values mandated by [DSP0223](#).

1664 7.4 Optional features of the CIM-RS protocol

1665 This subclause defines optional features for the implementation of the CIM-RS protocol.

1666 7.4.1 Entity tagging feature

1667 Implementation of the entity tagging feature in servers and clients provides for verifying the basis of
1668 resource modifications and thus for improved consistency control in instance modifications (see 7.3.7)
1669 and for improved cache control (see 7.3.3).

1670 Implementation of the entity tagging feature is optional for clients and servers, independently.

1671 Implementation of the entity tagging feature in a server is indicated through the "entitytagging" attribute in
1672 the server entry point resource (see 7.12).

1673 7.4.2 Continue on error feature

1674 Implementation of the continue on error feature in servers provides clients with the possibility to request
1675 continuation of a sequence of paged retrievals in case of error. For details on paged retrieval, see 7.3.8.

1676 Implementation of the continue on error feature is optional for clients and servers, independently.

1677 Implementation of the continue on error feature in a server is indicated through the "continueonerror"
1678 attribute in the server entry point resource (see 7.12).

1679 7.5 Instance creation resource

1680 An instance creation resource represents the ability to create instance resources.

1681 As defined in 7.14, a server exposes one instance creation resource for each namespace that is
1682 supported for access by the CIM-RS protocol; its resource identifier is available through the "creation"
1683 attribute of the corresponding entry of the "namespaces" array attribute of the server entry point resource
1684 (see 7.11).

1685 7.5.1 POST

1686 **Purpose:** Creates an instance resource

1687 **HTTP method:** POST

1688 **Target resource:** Instance creation resource (see 7.5)

1689	Query parameters:	<code>\$class</code>
1690	Request headers:	Host, Content-Length, Content-Type, X-CIMRS-Version
1691	Request payload:	Instance (see 7.6.1), without the "self" and "methods" attributes
1692	Response headers (success):	Date, Location, X-CIMRS-Version
1693	Response payload (success):	None
1694	Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
1695	Response payload (failure):	ErrorResponse (see 7.3.6)
1696	Requirement:	Mandatory (class specific)
1697	Description:	
1698		The HTTP POST method on an instance creation resource creates an instance of the specified class
1699		in the namespace of the targeted instance creation resource. The initial property values for the new
1700		instance are defined in an instance representation in the payload. On return, the Location header
1701		specifies the resource identifier of the newly created instance.
1702		The target resource identifier for this operation is specific to a namespace and can be obtained
1703		through the "creation" attribute of the corresponding entry of the "namespaces" array attribute of the
1704		server entry point resource (see 7.12). The entry for the desired namespace can be selected upfront
1705		by inspecting its "name" attribute. The desired class is specified as query parameter <code>\$class</code> (see
1706		6.5.1); it is required to be specified. If it is not specified, the server shall fail the operation with HTTP
1707		status code 404 "Not Found".
1708		The new instance shall have a creation class that is the class specified in the <code>\$class</code> query
1709		parameter in the namespace of the targeted instance creation resource.
1710		The set of properties to be initialized in the new instance by the server is the set of all properties
1711		exposed by the creation class.
1712		Properties specified in the Instance payload element represent client-supplied initial values for the
1713		new instance.
1714		Properties specified in the Instance payload element that are not properties exposed by the creation
1715		class shall cause the server to fail the operation with HTTP status code 403 "Forbidden". Properties
1716		specified in the Instance payload element that are not client-initializable shall cause the server to fail
1717		the operation with HTTP status code 403 "Forbidden".
1718		Client-initializable properties shall be initialized as specified for the property in the Instance payload
1719		element (including initializing the property to Null), or if the property is not specified in the Instance
1720		payload element, to the class-defined default value of the property, or to Null if no such default value
1721		is defined.
1722		Any other properties of the instance shall be initialized as defined by the implementation, taking into
1723		account any requirements on the initial values defined in the model.
1724		If the resulting initial values would violate these requirements, the server shall fail the operation with
1725		HTTP status code 403 "Forbidden".
1726		The "self" link in the Instance payload element in the request message shall not be specified. If
1727		specified, the request shall be rejected with HTTP status code 400 "Bad Request".
1728		Any method invocation links in the Instance payload element in the request message shall not be
1729		specified. If specified, the request shall be rejected with HTTP status code 400 "Bad Request".

1730 On success, the entity body shall contain no payload element and the following HTTP status code
1731 shall be returned:

- 1732 • 201 "Created": The "Location" header field is set to the resource identifier of the newly
1733 created instance

1734 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
1735 the following HTTP status codes shall be returned:

- 1736 • 400 "Bad Request": Requirements on the request payload element were not satisfied (for
1737 example, "self" link or method invocation links were specified)
- 1738 • 403 "Forbidden": Properties specified in the Instance payload element are not client-
1739 initializable, are not properties exposed by the creation class of the new instance, or the
1740 resulting initial values would violate requirements defined in the model
- 1741 • 404 "Not Found": Target instance creation resource does not exist, for example because
1742 the \$class query parameter is not specified, or because it specifies a non-existing class
- 1743 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
1744 method (see [RFC2616](#))

1745 Example HTTP conversation (using JSON):

1746 Request:

```
1747 POST /cimrs/root%2Fcimv2/create?$class=ACME_RegisteredProfile HTTP/1.1
1748 Host: server.acme.com:5988
1749 Content-Length: XXX
1750 Content-Type: application/json;version=1.0.0
1751 X-CIMRS-Version: 1.0.0
1752
1753 {
1754   "kind": "instance",
1755   "class": "ACME_RegisteredProfile",
1756   "properties": {
1757     "RegisteredName": "Fan",
1758     "RegisteredOrganization": 2,
1759     "RegisteredVersion": "1.1.0"
1760   }
1761 }
```

1762 Response:

```
1763 HTTP/1.1 201 Created
1764 Date: Fri, 11 Nov 2011 10:11:00 GMT
1765 Location: http://server.acme.com:5988/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMT
1766 F%3AFan%3A1.1.0
1767 X-CIMRS-Version: 1.0.1
```

1768 NOTE: The key property InstanceID is not provided in the request, since key property values are determined
1769 by the server. Other properties of the class (for example, Caption or Description) are initialized to their class-
1770 defined default values, or to Null.

1771 7.6 Instance resource

1772 An instance resource represents a managed object in the managed environment.

1773 Because CIM-RS is model-neutral, it defines how instances are exposed as instance resources. A model
 1774 defines how managed objects are modeled as instances, by defining classes. In combination, this defines
 1775 how managed objects are represented as REST instance resources. For details, see 5.5.

1776 **7.6.1 Instance payload element**

1777 An Instance payload element is the representation of an instance resource (and thus, of a managed
 1778 object in the managed environment) in the protocol.

1779 Unless otherwise constrained, an Instance payload element shall have the attributes defined in Table 8.

1780 **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

1781 The following requirements apply to the child attributes of the "properties" attribute, if present:

- 1782 • the "name" and "value" child attributes shall be present
- 1783 • the "type" child attribute shall be present if the payload representation supports the
 1784 representation of the CIM datatype in element values, and shall be omitted otherwise

1785 The following requirements apply to the child attributes of the "methods" attribute, if present:

- 1786 • the "name" and "uri" child attributes shall be present

1787 **7.6.2 DELETE**

1788 **Purpose:** Deletes an instance resource

1789 **HTTP method:** DELETE

1790 **Target resource:** Instance resource (see 7.6)

1791 **Query parameters:** None

1792 **Request headers:** Host, X-CIMRS-Version

1793 **Request payload:** None

1794 **Response headers (success):** Date, X-CIMRS-Version

- 1795 **Response payload (success):** None
- 1796 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version
- 1797 **Response payload (failure):** ErrorResponse (see 7.3.6)
- 1798 **Requirement:** Mandatory (class specific)
- 1799 **Description:**
- 1800 The HTTP DELETE method on an instance resource deletes the instance resource.
- 1801 On success, the entity body shall contain no payload element and the following HTTP status code
- 1802 shall be returned:
- 1803 • 204 "No Content"
- 1804 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
- 1805 the following HTTP status codes shall be returned:
- 1806 • 404 "Not Found": Target instance resource does not exist
 - 1807 • any other 4xx (client error) or 5xx (server error) HTTP status code permissible for this
 - 1808 HTTP method (see [RFC2616](#))
- 1809 **Example HTTP conversation (using JSON):**
- 1810 Request:
- ```
1811 DELETE /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0 HTTP/1.1
1812 Host: server.acme.com:5988
1813 X-CIMRS-Version: 1.0.0
```
- 1814 Response:
- ```
1815 HTTP/1.1 204 No Content
1816 Date: Fri, 11 Nov 2011 10:11:00 GMT
1817 X-CIMRS-Version: 1.0.1
```
- 1818 **7.6.3 GET**
- 1819 **Purpose:** Retrieves an instance resource
- 1820 **HTTP method:** GET
- 1821 **Target resource:** Instance resource (see 7.6)
- 1822 **Query parameters:** \$expand, \$refer, \$properties, \$methods, \$max,
- 1823 \$continueonerror, \$pagingtimeout
- 1824 **Request headers:** Host, Accept, X-CIMRS-Version
- 1825 **Request payload:** None
- 1826 **Response headers (success):** Date, Content-Length, Content-Type, ETag, X-CIMRS-Version
- 1827 **Response payload (success):** Instance (see 7.6.1)
- 1828 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version
- 1829 **Response payload (failure):** ErrorResponse (see 7.3.6)

1830 **Requirement:** Mandatory (class specific)

1831 **Description:**

1832 The HTTP GET method on an instance resource retrieves a representation of the specified instance
1833 resource.

1834 For details on the effects of the query parameters on the returned Instance payload element, see the
1835 descriptions of these query parameters in 6.5.

1836 **EXPERIMENTAL**

1837 Note that the returned Instance payload element may have navigation properties or expanded
1838 references as a result of using the `$expand` or `$refer` query parameters, as described in 5.6. Any
1839 collections in these navigation properties or expanded references may be paged (see 7.3.8), and the
1840 query parameters related to paged retrieval apply to those collections.

1841 **EXPERIMENTAL**

1842 On success, the entity body shall contain an Instance payload element (see 7.6.1) and one of the
1843 following HTTP status codes shall be returned:

- 1844 • 200 "OK": The entity body contains the response payload element
- 1845 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
1846 empty. This status code can only occur if the server supports conditional requests and the
1847 client has requested a conditional request

1848 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
1849 the following HTTP status codes shall be returned:

- 1850 • 404 "Not Found": Target instance resource does not exist
- 1851 • any other 4xx (client error) or 5xx (server error) HTTP status code permissible for this
1852 HTTP method (see [RFC2616](#))

1853 **Example HTTP conversation (using JSON):**

1854 Request:

```
1855 GET /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0 HTTP/1.1
1856 Host: server.acme.com:5988
1857 Accept: application/json;version=1.0
1858 X-CIMRS-Version: 1.0.0
```

1859 Response:

```
1860 HTTP/1.1 200 OK
1861 Date: Fri, 11 Nov 2011 10:11:00 GMT
1862 Content-Length: XXX
1863 Content-Type: application/json;version=1.0.1
1864 X-CIMRS-Version: 1.0.1
1865
1866 {
1867   "kind": "instance",
1868   "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0",
1869   "class": "ACME_RegisteredProfile",
```

```

1870     "properties": {
1871         "InstanceID": "DMTF:Fan:1.1.0",
1872         "RegisteredName": "Fan",
1873         "RegisteredOrganization": 2,
1874         "RegisteredVersion": "1.1.0",
1875         . . .
1876     },
1877     "methods": {
1878         "GetCentralInstances": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3
1879 A1.1.0/GetCentralInstances"
1880     }
1881 }

```

1882 7.6.4 PUT

1883	Purpose:	Modifies an instance resource (partially or fully)
1884	HTTP method:	PUT
1885	Target resource:	Instance resource (see 7.6)
1886	Query parameters:	\$properties
1887	Request headers:	Host, Content-Length, Content-Type, If-Match (EXPERIMENTAL), X-
1888		CIMRS-Version
1889	Request payload:	Instance (see 7.6.1)
1890	Response headers (success):	Date, X-CIMRS-Version
1891	Response payload (success):	None
1892	Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
1893	Response payload (failure):	ErrorResponse (see 7.3.6)
1894	Requirement:	Mandatory (class specific)
1895	Description:	

1896 The HTTP PUT method on an instance resource sets some or all property values of the specified
 1897 instance resource.

1898 Partial modification of an instance is achieved by specifying the desired subset of properties in the
 1899 resource identifier using the \$properties query parameter (see 6.5.8). Since query parameters
 1900 are part of the address of a resource (see [RFC2616](#)), this approach performs a full replacement of
 1901 the resource representing the partial instance, satisfying the idempotency requirement for the PUT
 1902 method demanded by [RFC2616](#).

1903 If the \$properties query parameter is not specified, the set of properties to be set is the set of all
 1904 mutable properties of the target instance. If the \$properties query parameter is specified, the set
 1905 of properties to be set is the set of properties specified in the \$properties query parameter.
 1906 Properties specified in the \$properties query parameter that are not properties of the target
 1907 instance shall cause the server to fail the operation with HTTP status code 404 "Not Found".
 1908 Properties specified in the \$properties query parameter that are not mutable shall cause the
 1909 server to fail the operation with HTTP status code 403 "Forbidden".

- 1910 Properties specified in the Instance payload element that are not to be set as previously defined,
1911 shall be tolerated and ignored, even when they are not properties of the target instance.
- 1912 Mutable properties that are to be set as previously defined shall be set as specified for the property
1913 in the Instance payload element (including setting the property to Null), or if the property is not
1914 specified in the Instance payload element, to the class-defined default value of the property, or to
1915 Null if no such default value is defined.
- 1916 NOTE: This behavior for properties that are to be set but not specified in the Instance payload element is
1917 consistent with CIM-XML ([DSP0200](#)). In contrast, generic operations ([DSP0223](#)) requires that the property is set
1918 to Null in this case, even when a non-Null default value for the property is defined in the class.
- 1919 Requirements on mutability of properties can be defined in the model. Key properties are always
1920 unmutable.
- 1921 The "self" link in the Instance payload element in the request message is optional. If specified, it shall
1922 reference the same resource as the target resource identifier.
- 1923 Any method invocation links in the Instance payload element in the request message should not be
1924 specified. If specified, they shall be ignored by the server.

1925 EXPERIMENTAL

- 1926 In addition, a server shall cause the PUT method to fail with HTTP status code 409 "Conflict" if an If-
1927 Match header field is provided, and the entity tag provided as its value does not match the current
1928 entity tag of the resource. See 7.4.1 for more details on verifying the basis for resource
1929 modifications.

1930 EXPERIMENTAL

- 1931 On success, the entity body shall contain no payload element and the following HTTP status code
1932 shall be returned:
- 1933 • 204 "No Content"
- 1934 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
1935 the following HTTP status codes shall be returned:
- 1936 • 403 "Forbidden": A property specified in the `$properties` query parameter was
1937 unmutable
 - 1938 • 404 "Not Found": Target instance resource does not exist; or the `$properties` query
1939 parameter specifies properties that are not properties of the target instance
 - 1940 • 409 "Conflict": Verification of the basis for resource modifications was requested by
1941 specifying an If-Match header field, and the entity tag specified in the If-Match header field
1942 did not match the current entity tag of the resource
 - 1943 • any other 4xx (client error) or 5xx (server error) HTTP status code permissible for this
1944 HTTP method (see [RFC2616](#))

1945 Example HTTP conversation (using JSON) for the full replacement of an instance:

1946 Request:

```
1947 PUT /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0 HTTP/1.1
1948 Host: server.acme.com:5988
1949 Content-Length: XXX
1950 Content-Type: application/json;version=1.0.0
```

```

1951 X-CIMRS-Version: 1.0.0
1952
1953 {
1954   "kind": "instance",
1955   "class": "ACME_RegisteredProfile",
1956   "properties": {
1957     "RegisteredName": "Fan",
1958     "RegisteredOrganization": 2,
1959     "RegisteredVersion": "1.1.1",
1960     "Caption": "A changed caption"
1961   }
1962 }

```

1963 Response:

```

1964 HTTP/1.1 200 OK
1965 Date: Fri, 11 Nov 2011 10:11:00 GMT
1966 X-CIMRS-Version: 1.0.1

```

1967 NOTE: In this example, it is assumed that all provided properties are mutable. The mutable properties not provided
 1968 (for example, Description) are set to their class-defined default values or to Null. The value of the InstanceID key
 1969 property remains unchanged, since key properties are never mutable.

1970 **Example HTTP conversation (using JSON) for the partial replacement of an instance:**

1971 Request:

```

1972 PUT /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0?$properties=Regis
1973   teredVersion,Caption HTTP/1.1
1974 Host: server.acme.com:5988
1975 Content-Length: XXX
1976 Content-Type: application/json;version=1.0.0
1977 X-CIMRS-Version: 1.0.0
1978
1979 {
1980   "kind": "instance",
1981   "class": "ACME_RegisteredProfile",
1982   "properties": {
1983     "RegisteredVersion": "1.1.1",
1984     "Caption": "A changed caption"
1985   }
1986 }

```

1987 Response:

```

1988 HTTP/1.1 200 OK
1989 Date: Fri, 11 Nov 2011 10:11:00 GMT
1990 X-CIMRS-Version: 1.0.1

```

1991 NOTE: In this example, it is assumed that all provided properties are mutable. Only the RegisteredVersion and
 1992 Caption properties are set to their new values.

1993 7.7 Reference collection resource

1994 A reference collection resource represents an order-preserving list of references to instance resources.

1995 **7.7.1 ReferenceCollection payload element**

1996 A ReferenceCollection payload element is the representation of a reference collection resource in the
 1997 protocol.

1998 Unless otherwise constrained, a ReferenceCollection payload element shall have the attributes defined in
 1999 Table 9.

2000 **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

2001 **7.7.2 GET**

2002 **Purpose:** Retrieves a reference collection resource

2003 **HTTP method:** GET

2004 **Target resource:** Reference collection resource (see 7.7)

2005 **Query parameters:** \$max, \$continueonerror, \$pagingtimeout

2006 **Request headers:** Host, Accept, X-CIMRS-Version

2007 **Request payload:** None

2008 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

2009 **Response payload (success):** ReferenceCollection (see 7.7.1)

2010 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2011 **Response payload (failure):** ErrorResponse (see 7.3.6)

2012 **Requirement:** Mandatory (class specific)

2013 **Description:**

2014 The HTTP GET method on a reference collection resource retrieves a representation of the specified
 2015 reference collection resource.

- 2016 The target resource identifier for this operation is typically discovered from the "next" attribute of
2017 reference collections that are returned in paged mode (see 7.3.8).
- 2018 For details on the effects of the query parameters on the returned ReferenceCollection payload
2019 element, see the descriptions of these query parameters in 6.5.
- 2020 Any retrieval of a reference collection may be paged (see 7.3.8).
- 2021 On success, the entity body shall contain a ReferenceCollection payload element (see 7.8.1) and
2022 one of the following HTTP status codes shall be returned:
- 2023 • 200 "OK": The entity body contains the response payload element
 - 2024 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
2025 empty. This status code can only occur if the server supports conditional requests and the
2026 client has requested a conditional request
- 2027 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
2028 the following HTTP status codes shall be returned:
- 2029 • 404 "Not Found": Target reference collection resource does not exist. This includes the
2030 case where paged retrieval is used and the sequence of paged retrievals has been closed
2031 by the server
 - 2032 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
2033 method (see [RFC2616](#))

2034 Example HTTP conversation (using JSON):

2035 Request:

```
2036 GET /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/refer/ACME_Elemen
2037 tConformsToProfile.ManagedElement/part/2 HTTP/1.1
2038 Host: server.acme.com:5988
2039 Accept: application/json;version=1.0
2040 X-CIMRS-Version: 1.0.0
```

2041 Response:

```
2042 HTTP/1.1 200 OK
2043 Date: Fri, 11 Nov 2011 10:11:00 GMT
2044 Content-Length: XXX
2045 Content-Type: application/json;version=1.0.1
2046 X-CIMRS-Version: 1.0.1
2047
2048 {
2049   "kind": "referencecollection",
2050   "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/refer/ACME
2051   _ElementConformsToProfile.ManagedElement/part/2",
2052   "class": "ACME_Fan",
2053   "references": [
2054     "/cimrs/root%2Fcimv2/ACME_Fan/fan11",
2055     "/cimrs/root%2Fcimv2/ACME_Fan/fan12"
2056   ]
2057 }
```

2058 In this example, a client had previously retrieved an ACME_RegisteredProfile instance for the DMTF Fan
 2059 Profile V1.1.0 and had requested the inclusion of a navigation property named
 2060 "ACME_ElementConformsToProfile.ManagedElement" by specifying
 2061 \$refer=ACME_ElementConformsToProfile.ManagedElement.

2062 The value of that navigation property is a reference collection, as it turns out, of ACME_Fan instances.
 2063 The server decided to return that reference collection in paged mode, and the first subset of 10 fan
 2064 references was part of the response to the original retrieval request. The representation of the collection
 2065 in that response included a "next" attribute for retrieving the next subset of the reference collection.

2066 What we see in the example above is the retrieval of that next subset, which happens to contain the
 2067 references to fans number 11 and 12, and no "next" attribute because this subset completed the
 2068 collection.

2069 **7.8 Instance collection resource**

2070 An instance collection resource represents an order-preserving list of instance resources, which are the
 2071 result of some operation such as instance enumeration or association traversal. An instance collection
 2072 resource in a response can be represented in its entirety, or in pages (see 7.3.8). If represented in its
 2073 entirety, the instance collection is embedded in the result and does not have a resource URI. If
 2074 represented in pages, the first page is embedded in the result and does not have a resource URI, and
 2075 any remaining pages have a resource URI specific to that page.

2076 **7.8.1 InstanceCollection payload element**

2077 An InstanceCollection payload element is the representation of an instance collection resource in the
 2078 protocol, both when represented in its entirety or when represented in pages.

2079 Unless otherwise constrained, an InstanceCollection payload element shall have the attributes defined in
 2080 Table 10.

2081 **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

2082 **7.8.2 GET**

2083 **Purpose:** Retrieves the next page of a paged instance collection resource

2084	HTTP method:	GET
2085	Target resource:	Page of an instance collection resource (see 7.8)
2086	Query parameters:	\$max
2087	Request headers:	Host, Accept, X-CIMRS-Version
2088	Request payload:	None
2089	Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version
2090	Response payload (success):	InstanceCollection (see 7.8.1)
2091	Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
2092	Response payload (failure):	ErrorResponse (see 7.3.6)
2093	Requirement:	Mandatory (class specific)
2094	Description:	
2095		The HTTP GET method on page of an instance collection resource retrieves a representation of the
2096		specified page of the overall instance collection.
2097		The target resource identifier for this operation is discovered from the "next" attribute of the previous
2098		page of the instance collection (see 7.3.8).
2099		For details on the effects of the query parameters on the returned InstanceCollection payload
2100		element, see the descriptions of these query parameters in 6.5.

2101 EXPERIMENTAL

2102 Note that the instances in the returned InstanceCollection payload element may have navigation
 2103 properties or expanded references as a result of using the \$expand or \$refer query parameters,
 2104 as described in 5.6. Any collections in these navigation properties or expanded references may be
 2105 paged (see 7.3.8), and the query parameters related to paged retrieval apply to those collections.

2106 EXPERIMENTAL

2107 Any retrieval of an instance collection may be paged (see 7.3.8).

2108 On success, the entity body shall contain an InstanceCollection payload element (see 7.8.1) and one
 2109 of the following HTTP status codes shall be returned:

- 2110 • 200 "OK": The entity body contains the response payload element
- 2111 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
 2112 empty. This status code can only occur if the server supports conditional requests and the
 2113 client has requested a conditional request

2114 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
 2115 the following HTTP status codes shall be returned:

- 2116 • 404 "Not Found": Target instance collection resource page does not exist. This includes
 2117 the case where paged retrieval is used and the sequence of paged retrievals has been
 2118 closed by the server
- 2119 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
 2120 method (see [RFC2616](#))

2121 **Example HTTP conversation (using JSON):**2122 **Request:**

```

2123 GET /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/ACME_ReferencedPr
2124 ofile/Antecedent HTTP/1.1
2125 Host: server.acme.com:5988
2126 Accept: application/json;version=1.0
2127 X-CIMRS-Version: 1.0.0

```

2128 **Response:**

```

2129 HTTP/1.1 200 OK
2130 Date: Fri, 11 Nov 2011 10:11:00 GMT
2131 Content-Length: XXX
2132 Content-Type: application/json;version=1.0.1
2133 X-CIMRS-Version: 1.0.1
2134
2135 {
2136   "kind": "instancecollection",
2137   "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/ACME_Refer
2138 encedProfile/Antecedent",
2139   "class": "ACME_RegisteredProfile",
2140   "instances": [
2141     {
2142       "kind": "instance",
2143       "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0",
2144       "class": "ACME_RegisteredProfile",
2145       "properties": {
2146         "InstanceID": "DMTF:Fan:1.1.0",
2147         "RegisteredName": "Fan",
2148         "RegisteredOrganization": 2,
2149         "RegisteredVersion": "1.1.0",
2150         . . . ,
2151         "ACME_ReferencedProfile": {
2152           "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/AC
2153 ME_ReferencedProfile",
2154           "Dependent": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0
2155 .0/ACME_ReferencedProfile/Dependent"
2156         }
2157       },
2158       "methods": {
2159         "GetCentralInstances": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AF
2160 an%3A1.1.0/GetCentralInstances"
2161       }
2162     },
2163     . . .
2164   ]
2165 }

```

2166 In this example, the operation traverses from a starting instance of class ACME_RegisteredProfile to the
 2167 set of instances associated through the ACME_ReferencedProfile association, specifically its Antecedent
 2168 end.

2169 The returned set of instances is again of class ACME_RegisteredProfile and has a navigation property
2170 named ACME_ReferencedProfile for navigating back.

2171 7.9 Instance enumeration resource

2172 An instance enumeration resource represents the ability to enumerate instances of a class (including
2173 subclasses) in a namespace of a server, returning them as an instance collection.

2174 As defined in 7.14, a server exposes one instance enumeration resource; its resource identifier is
2175 available through the "enumeration" attribute of the corresponding entry of the "namespaces" array
2176 attribute of the server entry point resource (see 7.11).

2177 7.9.1 GET

2178 **Purpose:** Enumerates instance resources by class

2179 **HTTP method:** GET

2180 **Target resource:** Instance enumeration resource (see 7.9)

2181 **Query parameters:** \$class, \$filter, \$expand, \$refer, \$properties, \$methods,
2182 \$max, \$continueonerror, \$pagingtimeout

2183 **Request headers:** Host, Accept, X-CIMRS-Version

2184 **Request payload:** None

2185 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

2186 **Response payload (success):** InstanceCollection (see 7.8.1)

2187 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2188 **Response payload (failure):** ErrorResponse (see 7.3.6)

2189 **Requirement:** Mandatory (class specific)

2190 Description:

2191 The HTTP GET method on an instance enumeration resource enumerates all instances of the
2192 specified class (including instances of subclasses) in the namespace of the targeted instance
2193 enumeration resource and returns an instance collection with representations of these instances.

2194 The target resource identifier for this operation is specific to a namespace and can be obtained
2195 through the "enumeration" attribute of the corresponding entry in the "namespaces" array attribute of
2196 the server entry point resource (see 7.11). The entry for the desired namespace can be selected
2197 upfront by inspecting its "name" attribute. The desired class is specified as query parameter \$class
2198 (see 6.5.1); it is required to be specified. If it is not specified, the server shall fail the operation with
2199 HTTP status code 404 "Not Found".

2200 For details on the effects of the query parameters on the returned InstanceCollection payload
2201 element, see the descriptions of these query parameters in 6.5.

2202 EXPERIMENTAL

2203 Note that the instances in the returned InstanceCollection payload element may have navigation
2204 properties or expanded references as a result of using the \$expand or \$refer query parameters,

2205 as described in 5.6. Any collections in these navigation properties or expanded references may be
 2206 paged (see 7.3.8), and the query parameters related to paged retrieval apply to those collections.

2207 EXPERIMENTAL

2208 Any retrieval of an instance collection may be paged (see 7.3.8)

2209 On success, the entity body shall contain an InstanceCollection payload element (see 7.8.1) and one
 2210 of the following HTTP status codes shall be returned:

- 2211 • 200 "OK": The entity body contains the response payload element. This includes the case
 2212 where the specified class and namespace exist, but the result set of instances is empty
- 2213 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
 2214 empty. This status code can only occur if the server supports conditional requests and the
 2215 client has requested a conditional request

2216 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
 2217 the following HTTP status codes shall be returned:

- 2218 • 404 "Not Found": Target instance enumeration resource does not exist, for example
 2219 because the `$class` query parameter is not specified, or because it specifies a non-
 2220 existing class. This includes the case where paged retrieval is used and the sequence of
 2221 paged retrievals has been closed by the server
- 2222 • any other 4xx (client error) or 5xx (server error) HTTP status code permissible for this
 2223 HTTP method (see [RFC2616](#))

2224 Example HTTP conversation:

2225 Request:

```
2226 GET /cimrs/root%2Fcimv2/enum?$class=ACME_System HTTP/1.1
2227 Host: server.acme.com:5988
2228 Accept: application/json;version=1.0
2229 X-CIMRS-Version: 1.0.1
```

2230 Response:

```
2231 HTTP/1.1 200 OK
2232 Date: Fri, 11 Nov 2011 10:11:00 GMT
2233 Content-Length: XXX
2234 Content-Type: application/json;version=1.0.0
2235 X-CIMRS-Version: 1.0.0
2236
2237 {
2238   "kind": "instancecollection",
2239   "self": "/cimrs/root%2Fcimv2/enum?$class=ACME_System",
2240   "class": "ACME_System",
2241   "instances": [
2242     {
2243       "kind": "instance",
2244       "self": "/cimrs/root%2Fcimv2/ACME_ComputerSystem/sys1",
2245       "class": "ACME_ComputerSystem",
2246       "properties": {
2247         "InstanceID": "sys1",
```

```

2248     "Name": "sys1",
2249     . . .
2250   },
2251   "methods": {
2252     "RequestStateChange": "/cimrs/root%2Fcimv2/ACME_ComputerSystem/sys1/Request
2253 StateChange"
2254   }
2255 },
2256 . . .
2257 ]
2258 }
    
```

2259 NOTE: This example assumes that ACME_ComputerSystem is a subclass of ACME_System.

2260 **7.10 Method invocation resource**

2261 A method invocation resource represents the ability to invoke a method defined in a class (static or non-
 2262 static). Non-static methods can be invoked on instances, using the method invocation resources available
 2263 through the "methods" attribute of an instance resource (see 7.6). Static methods can be invoked on
 2264 classes, using the method invocation resources available through the "staticmethods" attribute of the
 2265 corresponding entry of the "namespaces" array attribute of the server entry point resource (see 7.12).

2266 **7.10.1 MethodRequest payload element**

2267 A MethodRequest payload element is the representation of a request to invoke a method in the protocol.

2268 A MethodRequest payload element shall have the attributes defined in Table 11.

2269 **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

2270
 2271 The following requirements apply to the child attributes of the "parameters" attribute, if present:

- 2272 • the "name" and "value" child attributes shall be present
- 2273 • the "type" child attribute shall be present if the payload representation supports the
 2274 representation of the CIM datatype in element values, and shall be omitted otherwise

2275 **7.10.2 MethodResponse payload element**

2276 A MethodResponse payload element is the representation of the response of a method invocation in the
 2277 protocol.

2278 A MethodResponse payload element shall have the attributes defined in Table 12.

2279 **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

2280 The following requirements apply to the child attributes of the "returnvalue" attribute:

- 2281 • the "name" child attribute shall be omitted
- 2282 • the "value" child attribute shall be present
- 2283 • the "type" child attribute shall be present if the payload representation supports the
- 2284 representation of the CIM datatype in element values, and shall be omitted otherwise

2285 The following requirements apply to the child attributes of the "parameters" attribute, if present:

- 2286 • the "name" and "value" child attributes shall be present
- 2287 • the "type" child attribute shall be present if the payload representation supports the
- 2288 representation of the CIM datatype in element values, and shall be omitted otherwise

2289

2290 **7.10.3 POST**

- 2291 **Purpose:** Invokes a method (static or non-static)
- 2292 **HTTP method:** POST
- 2293 **Target resource:** Method invocation resource (see 7.10)
- 2294 **Query parameters:** None
- 2295 **Request headers:** Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
- 2296 **Request payload:** MethodRequest (see 7.10.1)
- 2297 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version
- 2298 **Response payload (success):** MethodResponse (see 7.10.2)
- 2299 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version
- 2300 **Response payload (failure):** ErrorResponse (see 7.3.6)
- 2301 **Requirement:** Mandatory (class specific)

2302 **Description:**

2303 The HTTP POST method on a method invocation resource invokes a method defined in a class
2304 (extrinsic method).

2305 The method can be static or non-static:

- 2306 • Non-static methods can be invoked on instances, using the method invocation links available
2307 through the "methods" attribute of an instance resource (see 7.6). A method invocation link for a
2308 non-static method is specific to the instance the method is invoked on, and to the method.
- 2309 • Static methods can be invoked on classes, using the method invocation links available through
2310 the "staticmethods" attribute of the corresponding entry of the "namespaces" array attribute of
2311 the server entry point resource (see 7.12). A method invocation link for a static method is
2312 specific to the class the method is invoked on, the namespace of the class, and to the method.

2313 On success, the entity body shall contain a MethodResponse payload element (see 7.10.2) and one
2314 of the following HTTP status codes shall be returned:

- 2315 • 200 "OK": The entity body contains the response payload element

2316 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
2317 the following HTTP status codes shall be returned:

- 2318 • 404 "Not Found": Target method invocation resource does not exist
- 2319 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
2320 method (see [RFC2616](#))

2321 Note that the ErrorResponse payload element used on failure cannot represent method output
2322 parameters or a method return value.

2323 **Example HTTP conversation (using JSON) for invocation of non-static method:**2324 **Request:**

```
2325 POST /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentralInstan
2326 ces HTTP/1.1
2327 Host: server.acme.com:5988
2328 Accept: application/json;version=1.0
2329 Content-Length: XXX
2330 Content-Type: application/json;version=1.0.0
2331 X-CIMRS-Version: 1.0.0
2332
2333 {
2334   "kind": " methodrequest",
2335   "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentral
2336 Instances",
2337   "method": "GetCentralInstances",
2338   "parameters": {
2339     "MaxNumber": 1000
2340   }
2341 }
```

2342 **Response:**

```
2343 HTTP/1.1 200 OK
2344 Date: Fri, 11 Nov 2011 10:11:00 GMT
```

```

2345 Content-Length: XXX
2346 Content-Type: application/json;version=1.0.1
2347 X-CIMRS-Version: 1.0.1
2348
2349 {
2350   "kind": " methodresponse",
2351   "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentral
2352 Instances",
2353   "method": "GetCentralInstances",
2354   "returnvalue": 0,
2355   "parameters": {
2356     "ActualNumber": 25
2357   }
2358 }
    
```

2359 **7.11 Listener destination resource**

2360 A listener destination resource in a listener represents the ability to deliver an indication to the listener.

2361 NOTE: Listener destination resources in listeners should not be confused with modeled objects in servers that may
 2362 are also called "listener destinations" in some models (for example, in the event model of the CIM Schema), but
 2363 merely describe the information in the server about the location of the listener.

2364 **7.11.1 IndicationDeliveryRequest payload element**

2365 An IndicationDeliveryRequest payload element is the representation of a request to deliver an indication
 2366 to a listener in the protocol.

2367 An IndicationDeliveryRequest payload element shall have the attributes defined in Table 13.

2368 **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

2369

2370 **7.11.2 POST**

2371 **Purpose:** Delivers an indication to a listener

2372 **HTTP method:** POST

2373 **Target resource:** Listener destination resource (see 7.11)

2374 **Query parameters:** None

2375 **Request headers:** Host, Accept, Content-Length, Content-Type, X-CIMRS-Version

2376 **Request payload:** IndicationDeliveryRequest (see 7.11.1)

2377 **Response headers (success):** Date, X-CIMRS-Version

2378 **Response payload (success):** None

2379 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2380 **Response payload (failure):** ErrorResponse (see 7.3.6)

2381 **Requirement:** Mandatory

2382 **Description:**

2383 The HTTP POST method on a listener destination resource delivers an indication to the listener
2384 specified in that resource.

2385 For implementations supporting the event model defined in the CIM Schema published by DMTF, the
2386 target resource identifier for this operation is the value of the Destination property of
2387 CIM_ListenerDestination instances that indicate the CIM-RS protocol in their Protocol property. For
2388 details, see the *DMTF Indications Profile* ([DSP1054](#)).

2389 On success, the entity body shall contain no payload element and one of the following HTTP status
2390 codes shall be returned:

- 2391 • 200 "OK"

2392 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
2393 the following HTTP status codes shall be returned:

- 2394 • 404 "Not Found": Target listener destination resource does not exist
- 2395 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
2396 method (see [RFC2616](#))

2397 **Example HTTP conversation (using JSON):**

2398 Request:

```
2399 POST /cimrs/dest1 HTTP/1.1
2400 Host: listener.acme.com:5988
2401 Accept: application/json;version=1.0
2402 Content-Length: XXX
2403 Content-Type: application/json;version=1.0.0
2404 X-CIMRS-Version: 1.0.1
2405
2406 {
2407   "kind": "indicationdeliveryrequest",
2408   "self": "/cimrs/dest1",
2409   "indication": {
2410     "kind": "instance",
2411     "class": "ACME_AlertIndication",
2412     "properties": {
2413       "AlertType": 4,
2414       "PerceivedSeverity": 5,
2415       "ProbableCause": 42,
2416       "Message": "BOND0007: Some error happened, rc=23.",
2417       "MessageArguments": [ "23" ],
2418       "MessageID": "BOND0007",
```

```

2419     "OwningEntity": "ACME"
2420   }
2421 }
2422 }
    
```

2423 Response:

```

2424 HTTP/1.1 204 No Content
2425 Date: Fri, 11 Nov 2011 10:11:00 GMT
2426 X-CIMRS-Version: 1.0.0
    
```

2427 7.12 Server entry point resource

2428 A server entry point resource describes protocol-level capabilities of a server, and provides a starting
 2429 point for discovering further resources in the server.

2430 The representation of the server entry point resource provides some server capabilities, the list of
 2431 namespaces for which the server supports the CIM-RS protocol, and resource identifiers of resources that
 2432 provide for performing operations:

- 2433 • instance enumeration resource: A HTTP GET (see 7.9.1) on this resource enumerates all
 2434 instances of a given class in the namespace of this resource. The namespace is implied from
 2435 this resource. The class is specified by the client using the `$class` query parameter (see
 2436 6.5.1).
- 2437 • instance creation resource: A HTTP POST (see 7.5.1) on this resource creates an instance of a
 2438 given class in the namespace of this resource (and thus the corresponding managed object).
 2439 The namespace is implied from this resource. The class is specified by the client using the
 2440 `$class` query parameter (see 6.5.1).
- 2441 • method invocation resources for static methods: A HTTP POST (see 7.10.3) on such a resource
 2442 invokes a static method on a class in a namespace. Class, method and namespace are implied
 2443 from this resource, and are also specified in the server entry point resource.

2444 Clients need to know class and namespace of some entry point instance(s) of the model(s) they want to
 2445 interact with, to get beyond this server entry point, and can use the instance enumeration resource to
 2446 retrieve these instances.

2447 7.12.1 ServerEntryPoint payload element

2448 A ServerEntryPoint payload element is the representation of a server entry point resource in the protocol.

2449 A ServerEntryPoint payload element shall have the attributes defined in Table 14.

2450 **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

Attribute name	Payload datatype	Requirement	Description
defaultpaging timeout	Integer	Mandatory	indicates the default paging timeout of the server. For details on paged retrieval, see 7.3.8
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

2451 Each entry in the "namespaces" array attribute shall have the child attributes defined in Table 15.

2452 **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)

2453 **7.12.2 GET**

2454 **Purpose:** Retrieves the entry point resource of a server

- 2455 **HTTP method:** GET
- 2456 **Target resource:** Server entry point resource (see 7.12)
- 2457 **Query parameters:** None
- 2458 **Request headers:** Host, X-CIMRS-Version
- 2459 **Request payload:** None
- 2460 **Response headers (success):** Date, X-CIMRS-Version
- 2461 **Response payload (success):** ServerEntryPoint (see 7.12.1)
- 2462 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version
- 2463 **Response payload (failure):** ErrorResponse (see 7.3.6)
- 2464 **Requirement:** Mandatory
- 2465 **Description:**
- 2466 The HTTP GET method on a server entry point resource retrieves a representation of the specified
 2467 server entry point resource. The returned ServerEntryPoint payload element describes protocol-level
 2468 capabilities of the server and its namespaces, such as supported protocol versions and supported
 2469 payload representations, as well as resource identifiers for discovering further resources in the
 2470 server and its namespaces.
- 2471 On success, the entity body shall contain a ServerEntryPoint payload element (see 7.12.1) and one
 2472 of the following HTTP status codes shall be returned:
- 2473 • 200 "OK": The entity body contains the response payload element
 - 2474 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
 2475 empty. This status code can only occur if the server supports conditional requests and the
 2476 client has requested a conditional request
- 2477 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
 2478 the following HTTP status codes shall be returned:
- 2479 • 404 "Not Found": Target server entry point resource does not exist
 - 2480 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
 2481 method (see [RFC2616](#))

2482 **Example HTTP conversation:**

2483 **Request:**

```
2484 GET /cimrs HTTP/1.1
2485 Host: server.acme.com:5988
2486 Accept: application/json;version=1.0
2487 X-CIMRS-Version: 1.0.0
```

2488 **Response:**

```
2489 HTTP/1.1 200 OK
2490 Date: Fri, 11 Nov 2011 10:11:00 GMT
2491 Content-Length: XXX
2492 Content-Type: application/json;version=1.0.1
2493 X-CIMRS-Version: 1.0.1
```

```

2494 {
2495   "kind": "serverentrypoint",
2496   "self": "/cimrs",
2497   "namespaces": [
2498     { "name": "interop",
2499       "enumeration": "/cimrs/interop/enum",
2500       "creation": "/cimrs/interop/create",
2501       "staticmethod": "/cimrs/interop/static",
2502       "protocolversions": [ "1.0.0", "1.0.1" ],
2503       "contenttypes": [
2504         "application/json;version=1.0.0",
2505         "application/json;version=1.0.1",
2506         "text/xml;version=1.0.0" ]
2507     },
2508     { "name": "root/cimv2",
2509       "enumeration": "/cimrs/root%2Fcimv2/enum",
2510       "creation": "/cimrs/root%2Fcimv2/create",
2511       "staticmethod": "/cimrs/root%2Fcimv2/static",
2512       "protocolversions": [ "1.0.0", "1.0.1" ],
2513       "contenttypes": [
2514         "application/json;version=1.0.0",
2515         "application/json;version=1.0.1",
2516         "text/xml;version=1.0.0" ]
2517     }
2518   ],
2519   "entitytagging": true,
2520   "pagedretrieval": true,
2521   "defaultpagingtimeout": 300,
2522   "minimumpagingtimeout": 1,
2523   "maximumpagingtimeout": 600,
2524   "continueonerror": true
2525 }
2526

```

2527 **7.13 Listener entry point resource**

2528 A listener entry point resource describes protocol-level capabilities of a listener.

2529 **7.13.1 ListenerEntryPoint payload element**

2530 A ListenerEntryPoint payload element is the representation of a listener entry point resource.

2531 A ListenerEntryPoint payload element shall have the attributes defined in Table 16.

2532 **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"

Attribute name	Payload datatype	Requirement	Description
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)
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)

2533 **7.13.2 GET**

2534 **Purpose:** Retrieves the entry point resource of a listener

2535 **HTTP method:** GET

2536 **Target resource:** Listener entry point resource (see 7.13)

2537 **Query parameters:** None

2538 **Request headers:** Host, X-CIMRS-Version

2539 **Request payload:** None

2540 **Response headers (success):** Date, X-CIMRS-Version

2541 **Response payload (success):** ListenerEntryPoint (see 7.13.1)

2542 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2543 **Response payload (failure):** ErrorResponse (see 7.3.6)

2544 **Requirement:** Mandatory

2545 **Description:**

2546 The HTTP GET method on a listener entry point resource retrieves a representation of the specified
 2547 listener entry point resource. The returned ListenerEntryPoint payload element describes protocol-
 2548 level capabilities of a listener, such as supported protocol versions and supported payload
 2549 representations.

2550 On success, the entity body shall contain a ListenerEntryPoint payload element (see 7.13.1) and one
 2551 of the following HTTP status codes shall be returned:

- 2552 • 200 "OK": The entity body contains the response payload element

- 2553 • 304 "Not Modified": The validators matched on a conditional request; the entity body is
2554 empty. This status code can only occur if the server supports conditional requests and the
2555 client has requested a conditional request

2556 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.6) and one of
2557 the following HTTP status codes shall be returned:

- 2558 • 404 "Not Found": Target listener entry point resource does not exist
- 2559 • any 4xx (client error) or 5xx (server error) HTTP status code permissible for this HTTP
2560 method (see [RFC2616](#))

2561 **Example HTTP conversation (server to listener):**

2562 Request:

```
2563 GET /cimrs HTTP/1.1
2564 Host: listener.acme.com:5988
2565 Accept: application/json;version=1.0
2566 X-CIMRS-Version: 1.0.1
```

2567 Response:

```
2568 HTTP/1.1 200 OK
2569 Date: Fri, 11 Nov 2011 10:11:00 GMT
2570 Content-Length: XXX
2571 Content-Type: application/json;version=1.0.0
2572 X-CIMRS-Version: 1.0.0
2573
2574 {
2575   "kind": "listenerentrypoint",
2576   "self": "/cimrs",
2577   "destinations": [ "/cimrs/dest1", "/cimrs/dest2" ],
2578   "protocolversions": [ "1.0.0" ],
2579   "contenttypes": [
2580     "application/json;version=1.0.0" ]
2581 }
```

2582 **7.14 CIM-RS resources to be exposed**

2583 This subclause summarizes which resources servers and listeners need to expose.

2584 **7.14.1 Resources exposed by a server**

2585 The following resources shall be exposed once by a server:

- 2586 • Server entry point resource (see 7.12)

2587 For each namespace that is supported for access by the CIM-RS protocol, the following resources shall
2588 be exposed by a server:

- 2589 • Instance enumeration resource (see 7.9)
- 2590 • Instance creation resource (see 7.5)
- 2591 • Method invocation resource (see 7.10) for static methods

2592 For each instance (including association instances) in each namespace that is supported for access by
2593 the CIM-RS protocol, the following resources shall be exposed by a server:

- 2594 • Instance resource (see 7.6)
- 2595 • Instance collection resources (see 7.8) and reference collection resources (see 7.7) that
2596 continue retrieval of such collections in paged mode. Note that the presence of these collections
2597 is highly dynamic
- 2598 • Method invocation resources (see 7.10); one for each non-static method that is exposed by the
2599 creation class of the instance and that is implemented

2600 7.14.2 Resources exposed by a listener

2601 The following resources shall be exposed once by a listener:

- 2602 • Listener entry point resource (see 7.13)

2603 For each listener destination supported by a listener, the following resources shall be exposed by the
2604 listener:

- 2605 • Listener destination resource (see 7.11)

2606 7.15 Other typical WBEM protocol functionality

2607 Certain functionality that is typical for a WBEM protocol or for systems management protocols in general
2608 does not have specific operations defined in the CIM-RS protocol, but can be performed by using other
2609 operations defined in the CIM-RS protocol, or discovery protocols, or the functionality of model-defined
2610 management interfaces accessible through the CIM-RS protocol. This subclause describes how a
2611 number of such functionalities can be performed.

2612 7.15.1 Server discovery

2613 WBEM servers can be discovered as described in clause 10.

2614 7.15.2 Discovery of server and listener entry point resources

2615 Once the IP address or hostname of a server or listener is known, the well-known resource identifier for
2616 its entry point resources can be constructed as described in 6.6, and using those, their entry point
2617 resources can be retrieved by performing the HTTP GET method on a server entry point resource (see
2618 7.12.2) and listener entry point resource (see 7.13.2), respectively.

2619 7.15.3 Namespace discovery

2620 The set of namespaces implemented by a server that support access through the CIM-RS protocol can
2621 be discovered from the "namespaces" attribute of the server entry point resource (see 7.12).

2622 7.15.4 Registered profile discovery

2623 The Profile Registration Profile ([DSP1033](#)) describes how to discover the management profiles to which a
2624 server advertises conformance, and from there, all further resources that are part of the functionality of a
2625 management profile. The management profiles to which a server advertises conformance can be
2626 discovered by enumerating instances of the CIM_RegisteredProfile class in the Interop namespace using
2627 the HTTP GET method on the instance enumeration resource for the Interop namespace (see 7.9.1).

2628 7.15.5 Schema inspection

2629 The schema definition (that is, class declarations and qualifier type declarations) including its meta-data
2630 in the form of qualifiers is expected to be accessible through a future "schema inspection model", using
2631 the existing operations defined in the CIM-RS protocol.

2632 7.15.6 Association traversal (EXPERIMENTAL)

2633 EXPERIMENTAL

2634 The CIM-RS protocol supports traversal of associations from a source instance to the association
2635 instances referencing the source instance, and to the instances associated with the source instance.
2636 There is no specific operation defined for this. Instead, it is performed by using the `$expand` (see 6.5.3)
2637 or `$refer` (see 6.5.9) query parameters to cause the inclusion of navigation properties for association
2638 traversal. For details on navigation properties, see 5.6.

2639 EXPERIMENTAL

2640 7.15.7 Indication subscription

2641 The CIM-RS protocol defines the HTTP POST method on listener destination resources (see 7.11.2) for
2642 the delivery of indications (that is, event notifications). However, it does not define any specific operations
2643 for performing other indication-related functions such as subscribing for indications, retrieving and
2644 managing indication filters and filter collections, or retrieving and managing listener destinations or
2645 indication services.

2646 Consistent with other WBEM protocols, the CIM-RS protocol leaves the definition of such functionality to a
2647 model-defined management interface, such as the *Indications Profile* ([DSP1054](#)).

2648 8 HTTP usage

2649 8.1 General requirements

2650 WBEM clients, servers, and listeners may support the use of HTTP for the CIM-RS protocol. The
2651 following applies if HTTP is supported:

- 2652 • Version 1.1 of HTTP shall be supported as defined in [RFC2616](#).
- 2653 • Version 1.0 or earlier of HTTP shall not be supported.

2654 WBEM clients, servers, and listeners shall support the use of HTTPS for the CIM-RS protocol. The
2655 following applies:

- 2656 • HTTPS shall be supported as defined in [RFC2818](#).
- 2657 • Within HTTPS, version 1.1 of HTTP shall be supported as defined in [RFC2616](#).

2658 NOTE 1 HTTPS should not be confused with Secure HTTP defined in RFC2660.

2659 8.2 Authentication requirements

2660 This subclause describes requirements and considerations for authentication between clients, servers,
2661 and listeners. Specifically, authentication happens from clients to servers for operation messages, and
2662 from servers to listeners for indication delivery messages.

2663 8.2.1 Operating without authentication

2664 WBEM clients, servers, and listeners may support operating without the use of authentication.

2665 This may be acceptable in environments such as physically isolated networks or between components on
2666 the same operating system.

2667 8.2.2 HTTP basic authentication

2668 HTTP basic authentication provides a rudimentary level of authentication, with the major weakness that
2669 the client password is part of the HTTP headers in unencrypted form.

2670 WBEM clients, servers, and listeners may support HTTP basic authentication as defined in [RFC2617](#).

2671 HTTP basic authentication may be acceptable in environments such as physically isolated networks,
2672 between components on the same operating system, or when the messages are encrypted by using
2673 HTTPS.

2674 8.2.3 HTTP digest authentication

2675 HTTP digest authentication verifies that both parties share a common secret without having to send that
2676 secret in the clear. Thus, it is more secure than HTTP basic authentication.

2677 WBEM clients, servers, and listeners should support HTTP digest authentication as defined in [RFC2617](#).

2678 8.2.4 Other authentication mechanisms

2679 WBEM clients, servers, and listeners may support authentication mechanisms not covered by [RFC2617](#).
2680 One example of such a mechanism is public key certificates as defined in [X.509](#).

2681 8.3 Message encryption requirements

2682 Encryption of HTTP messages can be supported by the use of HTTPS and its secure sockets layer.

2683 It is important to understand that authentication and encryption of messages are separate issues:
2684 Encryption of messages requires the use of HTTPS, while the authentication mechanisms defined in 8.2
2685 can be used with both HTTP and HTTPS.

2686 The following requirements apply to clients, servers, and listeners regarding the secure sockets layer
2687 used with HTTPS:

- 2688 • TLS 1.0 (also known as SSL 3.1) as defined in [RFC2246](#) shall be supported. Note that TLS 1.0
2689 implementations may be vulnerable when using CBC cipher suites
- 2690 • TLS 1.1 as defined in [RFC4346](#) should be supported
- 2691 • TLS 1.2 as defined in [RFC5246](#) should be supported
- 2692 • SSL 2.0 or SSL 3.0 shall not be supported because of known security issues in these versions

2693 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
2694 time of publication of this standard, it is expected that support for TLS 1.1 and TLS 1.2 is still not
2695 pervasive; therefore TLS 1.0 has been chosen as a minimum despite its known security issues.

2696 [RFC5246](#) describes in Appendix E "Backward Compatibility" how the secure sockets layer can be
2697 negotiated.

2698 The following requirements apply to clients, servers, and listeners regarding the cipher suites used with
2699 HTTPS:

- 2700 • The TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value 0x0013)
2701 shall be supported when using TLS 1.0. Note that [RFC2246](#) defines this cipher suite to be
2702 mandatory for TLS 1.0
- 2703 • The TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value 0x000A) shall
2704 be supported when using TLS 1.1. Note that [RFC4346](#) defines this cipher suite to be mandatory
2705 for TLS 1.1
- 2706 • The TLS_RSA_WITH_AES_128_CBC_SHA cipher suite (hexadecimal value 0x002F) shall be
2707 supported when using TLS 1.2. Note that [RFC5246](#) defines this cipher suite to be mandatory for
2708 TLS 1.2
- 2709 • The TLS_RSA_WITH_AES_128_CBC_SHA256 cipher suite (hexadecimal value 0x003C)
2710 should be supported when using TLS 1.2, in order to meet the transition to a security strength of
2711 112 bits (guidance is provided in NIST Special Publication 800-57 [\[NIST 800-57\]](#) and NIST
2712 Special Publication 800-131A [\[NIST 800-131A\]](#))
- 2713 • Any additional cipher suites may be supported

2714 8.4 HTTP header fields

2715 This subclause describes the use of HTTP header fields within the CIM-RS protocol, and it defines
2716 extension-header fields specific to the CIM-RS protocol.

2717 Any rules for processing header fields defined in [RFC2616](#) apply, particularly regarding whitespace
2718 stripping, line continuation, multiple occurrences of headers, and case insensitive treatment of field
2719 names.

2720 8.4.1 Accept

2721 The rules for the Accept request-header field defined in [RFC2616](#) apply. This subclause defines
2722 additional constraints on its use.

2723 The Accept header field may be provided on the request message of any operation that may return a
2724 response payload.

2725 If provided by a client, the Accept header field shall specify media types identifying CIM-RS payload
2726 representations (including version) that are supported by the client.

2727 The use of media ranges (that is, the asterisk character "*") in the type or subtype fields of the media type
2728 is not permitted in the CIM-RS protocol.

2729 NOTE: [RFC2616](#) permits the use of media ranges for the Accept header field. However, with the envisioned
2730 combinations of type and subtype values for CIM-RS, wildcarding based on type and subtype is not meaningful.

2731 If implemented, the "q" accept parameter shall be interpreted as a preference; interpreting it as a quality
2732 does not make sense for the CIM-RS protocol. Clients may provide the "q" accept parameter. Servers
2733 should implement the "q" accept parameter; if not implemented, it shall be tolerated if provided.

2734 NOTE: [RFC2616](#) does not specify recommendations for implementing the "q" accept parameter.

2735 NOTE: [RFC2616](#) distinguishes between general media type parameters (such as "version"), and accept
2736 parameters (such as "q"); the latter can be used only in the Accept header field, while general media type parameters
2737 can be considered part of the media type definition.

2738 Additional accept parameters (that is, beyond "q") are not permitted to be used in the Accept header field.
2739 For future extensibility, servers shall tolerate and ignore unknown additional accept parameters.

2740 If an Accept header field is provided, servers shall use one of the payload representations and version
2741 identified in the Accept header field for the response payload, considering the "q" accept parameter if
2742 implemented.

2743 The version specified in the "version" parameter of a media type shall be interpreted by the server as
2744 follows:

- 2745 • If an update version is included, it specifies the lowest acceptable update version (within the
2746 specified major version and acceptable minor versions); higher update versions shall be
2747 acceptable in addition. If no update version is included, the server shall assume a default of 0;
2748 that is, any update version is acceptable (within the specified major version and acceptable
2749 minor versions).
- 2750 • The minor version specifies the only acceptable minor version.
- 2751 • The major version specifies the only acceptable minor version.

2752 NOTE: These rules follow the usual DMTF convention for referencing versions: Update versions newer than the
2753 one specified are selected automatically if available, but newer minor (and of course, major) versions are selected
2754 automatically.

2755 If none of the payload representations identified in the Accept header field is supported by the server, it
2756 shall return HTTP status code 406 "not acceptable".

2757 NOTE: [RFC2616](#) only recommends returning HTTP status code 406 "not acceptable" in this case, but it does not
2758 require it.

2759 If no Accept header field is provided, servers may use any valid payload representation and version for
2760 the response payload.

2761 Within the constraints defined in this subclause, the payload representations specified in the Accept
2762 header field and the payload representations used in the response may change over time, even between
2763 the same combination of client and server. This implies that a server needs to evaluate the Accept header
2764 field (if present) on every request, even when the request is originated from the same client as before.

2765 Example:

```
2766 Accept: application/json; version=2.0,  
2767         application/json;version=1.0.1; q=0.5,  
2768         text/xml; version=1.0;q=0.2
```

2769 In this example, value of the Accept header field is distributed over multiple lines. The client
2770 expresses a preference for version 2.0.x ($x \geq 0$) of the CIM-RS JSON payload representation (by
2771 means of the default value of 1 for the "q" parameter), if that representation version is not available,
2772 then for version 1.0.x ($x \geq 1$) of the CIM-RS JSON representation, if that is not available then for
2773 version 1.0.x ($x \geq 0$) of the CIM-RS XML representation.

2774 8.4.2 Content-Type

2775 The rules for the Content-Type entity-header field defined in [RFC2616](#) apply. This subclause defines
2776 additional constraints on its use.

2777 As defined in [RFC2616](#), the Content-Type entity-header field shall be provided on the request message
2778 of any operation that passes a request payload and on the response message of any operation that
2779 returns a response payload.

2780 The Content-Type entity-header field shall specify the media type identifying the CIM-RS payload
2781 representation and version that is used for the content of the entity body. The "version" parameter of the
2782 media type shall include the major, minor and update version indicators.

2783 8.4.3 ETag (EXPERIMENTAL)

2784 EXPERIMENTAL

2785 The rules for the ETag response-header field defined in [RFC2616](#) apply. This subclause defines
2786 additional constraints on its use.

2787 The ETag response-header field shall be provided in the response to a HTTP GET method on an
2788 instance resource (see 7.6.3), if the entity tagging feature (see 7.4.1) is implemented by the server.

2789 In this case, the ETag response-header field shall be specified using the following format (defined in
2790 ABNF):

```
2791 ETag = "ETag" WS ":" entity-tag
```

2792 where `entity-tag` is a suitable entity tag as defined in [RFC2616](#), and `WS` is whitespace as defined in
2793 subclause "ABNF usage conventions". In models based on the CIM Schema published by DMTF, the
2794 Generation property defined in class `CIM_ManagedElement` is targeted for that purpose.

2795 Otherwise, the ETag response-header field shall not be provided by a server.

2796 The ETag response-header field shall not be provided in any other responses.

2797 EXPERIMENTAL

2798 8.4.4 If-Match (EXPERIMENTAL)

2799 EXPERIMENTAL

2800 The rules for the If-Match request-header field defined in [RFC2616](#) apply. This subclause defines
2801 additional constraints on its use.

2802 The If-Match request-header field may be provided in the request of a HTTP PUT method on an instance
2803 resource (see 7.6.4), if the entity tagging feature (see 7.4.1) is implemented by the client and the server
2804 that returned the instance that is being modified, has implemented the entity tagging feature as well.

2805 If provided, the If-Match request-header field shall be specified using the following format for its field value
2806 (defined in ABNF):

```
2807 If-Match-value = entity-tag
```

2808 where `entity-tag` is the entity tag of the ETag header field of the retrieved representation of the
2809 instance resource that is the basis for the modification.

2810 The If-Match request-header field shall not be provided in any other requests.

2811 EXPERIMENTAL

2812 8.4.5 X-CIMRS-Version

2813 The CIM-RS protocol version is the version of this document, without any draft level. The X-CIMRS-
2814 Version extension-header field shall identify the CIM-RS protocol version to which the request or
2815 response conforms, using the following format for its field value (defined in ABNF):

2816 `X-CIMRS-Version-value = M "." N "." U`

2817 where *M* is the major version indicator, *N* is the minor version indicator, and *U* is the update version
 2818 indicator within the version. Each of these version indicator strings shall be a decimal representation of
 2819 the corresponding version indicator number without leading zeros. Note that each indicator version string
 2820 may include more than a single decimal digit.

2821 The X-CIMRS-Version extension-header field shall be included in any request and in any response.

2822 Example:

2823 `X-CIMRS-Version: 1.0.0`

2824 9 Payload representation

2825 CIM-RS payload representation specifications define how the abstract payload elements defined in this
 2826 document are encoded in the entity body of the HTTP messages used by the CIM-RS protocol. Such an
 2827 encoding format is termed a "*payload representation*" in this document.

2828 This clause defines requirements for payload representation specifications and for implementations of the
 2829 CIM-RS protocol that are related to payload representations.

2830 9.1 Internet media types

2831 The CIM-RS protocol uses Internet media types, as defined in section 3.7 of [RFC2616](#), for identifying the
 2832 payload representation of its abstract payload elements. This subclause defines requirements related to
 2833 media types used for the CIM-RS protocol.

2834 9.1.1 General

2835 CIM-RS payload representation specifications shall define a single media type that uniquely identifies a
 2836 payload representation across all payload representations listed in Table 18.

2837 It is recommended that any such media types be registered with IANA.

2838 Any media types used for the CIM-RS protocol shall identify the version of the payload representation
 2839 using a media type parameter named "version", as described in 9.1.2.1.

2840 Example of a media type that is valid for the CIM-RS protocol:

2841 `application/json; version=1.0`

2842 9.1.2 Media type parameters

2843 Table 17 defines parameters of media types used for the CIM-RS protocol. Parameters not listed in the
 2844 table are not permitted to be used. For future extensibility, consumers of media types shall tolerate and
 2845 ignore unknown media type parameters.

2846 **Table 17 – Media type parameters**

Parameter	Presence Requirement	Description
version	Mandatory	See 9.1.2.1.

2847 9.1.2.1 Parameter "version"

2848 The media type parameter named "version" shall identify the version of the payload representation
 2849 identified by the media type, using the following format for its value (defined in ABNF):

2850 `version-value = M ["." N ["." U]]`

2851 where *M* is the major version indicator, *N* is the minor version indicator, and *U* is the update version
2852 indicator within the version. Each of these version indicator strings shall be a decimal representation of
2853 the corresponding version indicator number without leading zeros. Note that each indicator version string
2854 may include more than a single decimal digit.

2855 Subclauses in this document that describe the usage of media types define additional requirements on
2856 the presence of the minor and update version indicators in the value of the "version" parameter.

2857 The semantics for these version indicators shall be the semantics defined by DMTF for its specification
2858 versions. The version indicators of payload representation specifications provided by third parties shall
2859 conform to that semantics.

2860 9.2 Payload element representations

2861 CIM-RS payload representation specifications shall define a representation for each payload element
2862 listed in Table 4.

2863 The representations of these payload elements should be designed such that they can represent
2864 elements from any valid model without introducing restrictions, and such that there is no need to extend
2865 the payload representation specification if the model gets extended.

2866 Attributes of the payload elements defined in this document may be represented in any way in the
2867 payload representation. The attribute names stated in the descriptions of the payload elements in clause
2868 7 do not need to be retained in the payload representation. The payload datatypes stated in Table 5 do
2869 not need to correspond 1:1 to datatypes the representation format may use, as long as the value range of
2870 the attribute values can be correctly represented without any restrictions or loss of information.

2871 For example, in a JSON representation of an Instance payload element (see 7.6.1), all of the following
2872 options would be valid for representing the "self" attribute for resource identifier "/cimrs/machine/1234":

- 2873 • as a JSON attribute with the same name as the attribute of the abstract payload element:

```
2874 {  
2875   "self": "/cimrs/machine/1234",  
2876   . . .  
2877 }
```

- 2878 • as a JSON attribute with a different name as the attribute of the abstract payload element:

```
2879 {  
2880   "this": "/cimrs/machine/1234",  
2881   . . .  
2882 }
```

- 2883 • as an entry in a JSON array for links following the rel/href approach:

```
2884 {  
2885   "links": [  
2886     { "rel": "self",  
2887       "href": "/cimrs/machine/1234" },  
2888     . . .  
2889   ],  
2890   . . .  
2891 }
```

2892 **9.3 Payload representations**

2893 Table 18 lists known payload representations and requirements to implement them; payload
 2894 representations not listed in Table 18 may be implemented in addition.

2895 This table will be kept up to date in future versions of this document to include known payload
 2896 representations, in order to provide a basis on which the media type can be kept unique.

2897 **Table 18 – CIM-RS payload representations**

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

2898

2899 **10 Discovery requirements**

2900 The CIM-RS protocol has the following requirements related to discovery protocols:

2901 WBEM servers should implement the SLP discovery protocol, supporting the provisions set forth in
 2902 [DSP0205](#), supporting the SLP template defined in [DSP0206](#).

2903 The CIM-RS protocol has no requirements for supporting the discovery of listeners. Note that listeners are
 2904 HTTP servers.

2905 **11 Version compatibility**

2906 This clause defines the rules for version compatibility between WBEM clients and servers.

2907 Since HTTP is session-less, the general principle for determining version compatibility in the CIM-RS
 2908 protocol is that the version for the relevant layers of the CIM-RS protocol is included in all protocol
 2909 messages, allowing the receiving participant to determine whether it is able to support that version.

2910 The general principle for backwards compatibility (as further detailed in this clause) is that servers are
 2911 backwards compatible to clients; that is, servers of a particular version work with "older" versions of
 2912 clients.

2913 Version compatibility for the CIM-RS protocol is defined for the following protocol layers:

- 2914 • HTTP protocol (see 11.1)
- 2915 • CIM-RS protocol (see 11.2)
- 2916 • CIM-RS payload representation (see 11.3)

2917 A client and a server are version-compatible with each other only if they are compatible at each of these
 2918 three protocol layers.

2919 **11.1 HTTP protocol version compatibility**

2920 As defined in [RFC2616](#), every HTTP request and every HTTP response shall indicate the HTTP protocol
 2921 version to which the message format conforms.

2922 Since the CIM-RS protocol requires support for HTTP 1.1 (see 8.1), the backward compatibility rules for
 2923 supporting HTTP 1.0 and HTTP 0.9 as defined in section 19.6 (Compatibility with Previous Versions) of
 2924 [RFC2616](#) do not need to be followed in order to conform to the CIM-RS protocol.

2925 At this point, there is no HTTP version higher than 1.1 defined. Therefore, a client and a server are
2926 compatible w.r.t. the HTTP protocol version only if they both support HTTP 1.1.

2927 **11.2 CIM-RS protocol version compatibility**

2928 As defined in 8.4.5, every HTTP request and every HTTP response in the CIM-RS protocol shall indicate
2929 the CIM-RS protocol version to which the request or response conforms, by including the X-CIMRS-
2930 Version extension-header field. As defined in 8.4.5, the X-CIMRS-Version extension-header field
2931 identifies major, minor and update version of the CIM-RS protocol.

2932 A client and a server are compatible w.r.t. the CIM-RS protocol version only if the following condition is
2933 satisfied:

- 2934 • the major version of the server is equal to the major version of the client, and the minor version
2935 of the server is equal to or larger than the minor version of the client.

2936 The update version is not considered in this rule because new update versions (within the same major
2937 and minor version) are not supposed to introduce new functionality, so this rule allows clients and servers
2938 to be upgraded to conform to new update versions of the CIM-RS protocol independently of each other.

2939 **11.3 CIM-RS payload representation version compatibility**

2940 As defined in 9.1, the CIM-RS payload representation is identified using a media type whose "version"
2941 parameter identifies its major, minor and update version.

2942 A client and a server are compatible w.r.t. the version of a particular payload representation only if the
2943 following condition is satisfied:

- 2944 • the major version of the server is equal to the major version of the client, and the minor version
2945 of the server is equal to or larger than the minor version of the client.

2946 The update version is not considered in this rule because new update versions (within the same major
2947 and minor version) are not supposed to introduce new functionality, so this rule allows clients and servers
2948 to be upgraded to conform to new update versions of the payload representation independently of each
2949 other.

2950 **12 Conformance**

2951 This clause defines the criteria for WBEM clients, servers, and listeners to implement the CIM-RS
2952 protocol conformant to this document.

2953 WBEM clients, servers, and listeners implement the CIM-RS protocol conformant to this document only if
2954 they satisfy all provisions set out in this document.

2955 The terms client, server, and listener in this document refer to clients, servers, and listeners that are
2956 conformant to this document, without explicitly mentioning that.

ANNEX A (normative)

2957
2958
2959
2960

Common ABNF rules

2961 This annex defines common ABNF rules used throughout this document.

2962 `nonZeroDecimalDigit = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"`

2963 `decimalDigit = "0" / nonZeroDecimalDigit`

2964 `leadingZeros = 1*"0"`

2965 `positiveDecimalInteger = [leadingZeros] nonZeroDecimalDigit *decimalDigit`

2966 `nonNegativeDecimalInteger = [leadingZeros] ("0" / nonZeroDecimalDigit *decimalDigit)`

2967

ANNEX B (informative)

Mapping CIM-RS to generic operations

2972 This annex describes how CIM-RS is to be mapped to generic operations (see [DSP0223](#)). This mapping
2973 can be used when adding support for the CIM-RS protocol to CIM servers that internally support the
2974 semantics of generic operations either directly or indirectly through a (further) mapping.

2975 **B.1 URI composition**

2976 CIM-RS does not specify the structure of URIs. URIs are considered opaque to the client, leaving each
2977 server implementation free to structure them as necessary. However, there will be some units of
2978 information that the server must be able to infer from a particular URI, and be able to perform bidirectional
2979 lossless translations between the URI and the information units. The server is free to enable this
2980 translation as it sees fit. This might be done by encoding the information into the URI, or by keeping a
2981 cache of the information indexed by a short hash that is encoded into the URI, or by any other means.

2982 The subclauses below describe the units of information that must be represented in the URI of each
2983 resource type (see Table 2). Unless otherwise stated, units of information are represented in the path
2984 component of the URI, in a server-specific way. Some information units are represented in CIM-RS query
2985 parameters, so they should not additionally be represented in the path component. Note that query
2986 parameters in a URI are considered part of the resource address (see [RFC3986](#)).

2987 **B.1.1 Instance creation resource**

2988 This resource represents the ability to create instance resources in a particular CIM namespace (see 7.5).
2989 Its URI enables the server to identify:

- 2990 • CIM namespace in which the new instance is to be created;
- 2991 • The name of the creation class of the instance to be created (represented in the URI through
2992 the `$class` query parameter, see 6.5.1);
- 2993 • The type of the resource (in this case, an instance creation resource).

2994 **B.1.2 Instance resource**

2995 This resource represents a managed object in the managed environment, through a CIM instance (see
2996 7.6). Its URI enables the server to identify:

- 2997 • CIM namespace of the instance (this is also the namespace of its creation class);
- 2998 • Name of instance's creation class;
- 2999 • Key bindings of the instance (name/value pairs of all key properties);
- 3000 • The type of the resource (in this case, an instance resource).

3001 **B.1.3 Page of instance or reference collection resource from association traversal** 3002 **(EXPERIMENTAL)**

3003 **EXPERIMENTAL**

3004 An instance collection resource represents a collection of instance resources (see 7.8). A reference
3005 collection resource represents a collection of references to instance resources (see 7.7). Instance or

3006 reference collection resources representing the result of an association traversal from a source instance
 3007 do not have URIs; their representation is always embedded as the value of a navigation property (see
 3008 5.6) in the source instance. If such an instance or reference collection is returned using paging (see
 3009 7.3.8), the pages following the initial (embedded) part of the collection have URIs. The URI of such a
 3010 page enables the server to identify:

- 3011 • CIM namespace of the source instance;
- 3012 • Name of creation class of the source instance;
- 3013 • Key bindings of the source instance (name/value pairs of all key properties);
- 3014 • The relationship of the source instance to the result, represented in the URI through the
 3015 `$expand` (see 6.5.3) and `$refer` (see 6.5.9) query parameters;
- 3016 • Some information identifying the page in the overall result;
- 3017 • The type of the resource and kind of result (in this case, a page of an instance or reference
 3018 collection resource resulting from association traversal).

3019 EXPERIMENTAL

3020 B.1.4 Page of instance or reference collection resource from enumeration by class

3021 An instance collection resource represents a collection of instance resources (see 7.8). A reference
 3022 collection resource represents a collection of references to instance resources (see 7.7). Instance or
 3023 reference collection resources representing the result of an enumeration of instances of a given class do
 3024 not have URIs; their representation is returned in the protocol payload (see 7.9). If such an instance or
 3025 reference collection is returned using paging (see 7.3.8), the pages following the initial (payload) part of
 3026 the collection have URIs. The URI of such a page enables the server to identify:

- 3027 • CIM namespace of the given class and the instances in the result set;
- 3028 • Name of the given class;
- 3029 • Some information identifying the page in the overall result;
- 3030 • The type of the resource and kind of result (in this case, a page of an instance or reference
 3031 collection resource resulting from enumeration by class).

3032 B.1.5 Instance enumeration resource

3033 This resource represents the ability to enumerate instances of a given class (including instances of
 3034 subclasses) in a particular CIM namespace (see 7.9). Its URI enables the server to identify:

- 3035 • CIM namespace of the given class;
- 3036 • Name of the given class (represented in the URI through the `$class` query parameter, see
 3037 6.5.1);
- 3038 • The type of the resource (in this case, an instance enumeration resource).

3039 B.1.6 Static method invocation resource

3040 This resource represents the ability to invoke a static method upon a class that exposes that method (see
 3041 7.10). Its URI enables the server to identify:

- 3042 • CIM namespace of the class upon which the method is to be invoked;
- 3043 • Name of the class upon which the method is to be invoked;
- 3044 • Name of the method;

- The type of the resource (in this case, a static method invocation resource).

B.1.7 Non-static method invocation resource

This resource represents the ability to invoke a non-static method upon an instance whose creation class exposes that method (see 7.10). Its URI enables the server to identify:

- CIM namespace of the instance upon which the method is to be invoked;
- Name of the creation class of the instance upon which the method is to be invoked;
- Key bindings of the instance upon which the method is to be invoked (name/value pairs of all key properties);
- Name of the method;
- The type of the resource (in this case, a non-static method invocation resource).

B.1.8 Listener destination resource

This resource represents the ability to deliver an indication to a listener (see 7.11). Its URI enables the server to identify:

- The listener to which the indication is to be delivered;
- The type of the resource (in this case, a listener destination resource).

B.1.9 Server and listener entry point resources

This resource describes protocol-level capabilities of a server or listener, and provides a starting point for discovering further resources in the server. This is the only resource for which CIM-RS specifies the format of the resource. Its URI encodes the following information:

- The type of the resource (in this case, the server or listener entry point resource); this is specified to be: `/cimrs`

B.2 Query parameters

Specific query parameters can be used with multiple CIM-RS operation/resource pairs. Likewise, many input parameters are common between multiple generic operations, and are used consistently across those operations. With minor exceptions, the usage of any particular CIM-RS query parameter can be mapped directly to specific generic operation parameters, regardless of the CIM-RS operation/resource pair with which it is used.

Table B-1 defines the mapping of CIM-RS query parameters to generic operations input parameters.

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 (EXPERIMENTAL)		See B.2.1
\$max	MaxObjectCount	Directly equivalent

CIM-RS Query Parameter	Generic Operations Input Parameter	Mapping
\$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 (EXPERIMENTAL)		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)

3074 **B.2.1 Special handling for \$expand and \$refer query parameters (EXPERIMENTAL)**

3075 **EXPERIMENTAL**

3076 \$expand and \$refer direct the server to traverse associations or reference properties in the result set.
 3077 Each \$expand or \$refer specification indicates one association traversal path, composed of an
 3078 arbitrary number of association hops. Multiple paths may be specified in a single CIM-RS operation.

3079 \$expand and \$refer are permitted on CIM-RS operations which target a single instance or an instance
 3080 collection. For each single instance, or each instance in a collection targeted by the CIM-RS operation,
 3081 the server is directed to apply all \$expand and \$refer paths, thereby including the additional
 3082 information requested.

3083 The values supplied to \$expand and \$refer query parameters are formatted in the same way. For
 3084 either query parameter, the query parameter value is an association traversal path composed of an
 3085 arbitrary length sequence of alternating association classes and reference properties, delimited by the
 3086 period ('.') character. Each reference property within the path may have an optional class name to act as
 3087 a filter on the types of instances to be considered at that point in the association traversal. Likewise for
 3088 either query parameter, the association traversal path is applied to each instance targeted by the CIM-RS
 3089 operation, and a representation of the final element in that traversal path is added to the result set.

3090 The difference between \$expand and \$refer is in the representation of the returned element. In the
 3091 case of \$expand, the information returned is an instance collection representation of the terminal
 3092 navigation hop element. In the case of \$refer, the information returned is a reference collection of the
 3093 terminal navigation hop element.

3094 An implementation may do the following.

- 3095 1) Identify all association traversal paths identified in all \$expand and \$refer query parameters
- 3096 supplied to the current operation. Merge the paths into a tree representation, so that common

- 3097 early portions of the different traversal paths need not be redundantly traversed. In this way the
3098 instance targeted by the CIM-RS operation is applied to the root of the traversal tree, and the
3099 leaves of the traversal tree represent the results of the individual association traversal paths.
3100 Note that if some traversal paths are strict supersets of others, this will result in a situation
3101 where not all traversal paths end in leaf nodes of the traversal tree. For each instance targeted
3102 by the CIM-RS operation, the tree is traversed to identify and supply the additional information
3103 requested in the query parameters, as described in subsequent steps.
- 3104 2) When `$expand` or `$refer` is supplied for any CIM-RS operation, it will map to generic
3105 operations in a common fashion regardless of which CIM-RS operation was invoked. In any
3106 case, it is assumed that the CIM-RS operation being invoked will begin by obtaining an initial
3107 instance or instance collection. Once that instance or collection is obtained, the following
3108 generic operations mapping will be performed, using the initial instance or instance collection as
3109 the “working instance collection”.
- 3110 3) Obtain the initial association traversal element from the root of the traversal tree identified in
3111 step 1) above.
- 3112 4) For each Working Instance in the working instance collection, perform the following. If the
3113 current traversal tree node specifies both association class and reference, then perform a
3114 generic operations `OpenAssociatorPaths` operation; if only association class is given,
3115 perform a generic operations `OpenReferencePaths` operation. (See step 6) below for
3116 possible modifications to generic operations method being called.) In either case, the call is
3117 made with the following parameters:
- 3118 • `SourceInstancePath` is formed from:
 - 3119 – The CIM namespace (extracted from the Working Instance);
 - 3120 – The class name (extracted from the Working Instance);
 - 3121 – Key property name/value pairs (extracted from the Working Instance).
 - 3122 • `AssociationClassName` is extracted from the class name specified in the current
3123 traversal tree node.
 - 3124 • `AssociatedClassName` is set to NULL.
 - 3125 • `SourceRoleName` is set to NULL.
 - 3126 • `AssociatedRoleName` is set to the reference name obtained from the current traversal
3127 tree node, if reference name is present; if not present, `AssociatedRoleName` is set to
3128 NULL.
 - 3129 • `FilterQueryString` is set from the `$filter` query parameter as described in B.2.1.
 - 3130 • `FilterQueryLanguage` is set to "DMTF:FQL" (see C.2).
 - 3131 • `OperationTimeout` is set from the `$pagingtimeout` query parameter as described in
3132 Table B-1.
 - 3133 • `ContinueOnError` is set from the `$continueonerror` query parameter as described in
3134 Table B-1.
 - 3135 • `MaxObjectCount` is set from the `$max` query parameter as described in Table B-1.
- 3136 5) If the current traversal tree node contains sub-nodes, then perform N recursions into step 4)
3137 above, setting the “current traversal tree node” to each of the N traversal tree sub-nodes.
- 3138 6) Special case: if the current traversal tree node corresponds to a terminal node in a `$expand`
3139 query parameter, then entire instances must be obtained instead of only instance paths.
3140 Therefore:

- 3141 a) Call `OpenAssociatedInstancesWithPath` instead of `OpenAssociatorPaths`, or
- 3142 b) Call `OpenReferences` operation instead of `OpenReferencePaths`.
- 3143 c) In either case, the following parameters will be supplied to the generic operations method:
- 3144
- `IncludeClassOrigin` is set to `FALSE`.
 - `IncludedProperties` is set from the `$properties` query parameter as described in Table B-1.
 - `ExcludeSubclassProperties` is set to `FALSE`.

3148 EXPERIMENTAL

3149 B.3 Server operations

3150 This subclause describes a server's decision tree for how incoming CIM-RS operations are to be
3151 analyzed, identified, and mapped to generic operations: for each HTTP method, the server will examine
3152 its target URI. Based upon the server's defined URI structure, it will determine what type of resource is
3153 targeted, and will then determine which generic operations are to be invoked.

3154 The following subclauses describe each combination of HTTP method and resource type (and in some
3155 cases, multiple variants of the same resource type).

3156 B.3.1 POST instance creation resource

3157 This CIM-RS operation creates an instance resource (see 7.5.1).

3158 This CIM-RS operation directly maps to the generic operation `CreateInstance`.

3159 The input parameters for this generic operation are formed as follows:

- 3160
- the `ClassPath` parameter is formed from:
 - 3161 – the CIM namespace, which is formed from information units extracted from the target URI
 - 3162 of the HTTP request (see B.1.1)
 - 3163 – the class name, obtained from the `$class` query parameter in the target URI of the HTTP
 - 3164 request (see B.1.1)
 - the `InstanceSpecification` parameter is formed from the class name and from the
3165 `properties` attribute of the `Instance` payload element in the HTTP request (see 7.6.1)
 - 3166

3167 The output parameters of this generic operation are used as follows:

- the `InstancePath` parameter is used to form the URI in the `Location` header of the HTTP
3168 response
- 3169

3170 Restrictions: None.

3171 B.3.2 POST static method invocation resource

3172 This CIM-RS operation invokes a static method defined in a class (extrinsic method), upon a class (see
3173 7.10.3).

3174 This CIM-RS operation directly maps to the generic operation `InvokeStaticMethod`.

3175 The input parameters for this generic operation are formed as follows:

- 3176 • the `ClassPath` parameter is formed from CIM namespace and class name, which are formed
3177 from information units extracted from the target URI of the HTTP request (see B.1.6)
- 3178 • the `MethodName` parameter is formed from information units extracted from the target URI of
3179 the HTTP request (see B.1.6)
- 3180 • the `InParmValues` parameter is formed from the `parameters` attribute of the
3181 `MethodRequest` payload element in the HTTP request (see 7.10.1)

3182 The output parameters of this generic operation are used as follows:

- 3183 • the `OutParmValues` parameter is used to form the `parameters` attribute of the
3184 `MethodResponse` payload element in the HTTP response (see 7.10.2)
- 3185 • the `ReturnValue` parameter is used to form the `returnvalue` attribute of the
3186 `MethodResponse` payload element in the HTTP response (see 7.10.2)

3187 Restrictions: None.

3188 B.3.3 POST non-static method invocation resource

3189 This CIM-RS operation invokes a non-static method defined in a class (extrinsic method), upon an
3190 instance (see 7.10.3).

3191 This CIM-RS operation directly maps to the generic operation `InvokeMethod`.

3192 The input parameters for this generic operation are formed as follows:

- 3193 • the `InstancePath` parameter is formed from CIM namespace, class name and key bindings,
3194 which are all formed from information units extracted from the target URI of the HTTP request
3195 (see B.1.7)
- 3196 • the `MethodName` parameter is formed from information units extracted from the target URI of
3197 the HTTP request (see B.1.7)
- 3198 • the `InParmValues` parameter is formed from the `parameters` attribute of the
3199 `MethodRequest` payload element in the HTTP request (see 7.10.1)

3200 The output parameters of this generic operation are used as follows:

- 3201 • the `OutParmValues` parameter is used to form the `parameters` attribute of the
3202 `MethodResponse` payload element in the HTTP response (see 7.10.2)
- 3203 • the `ReturnValue` parameter is used to form the `returnvalue` attribute of the
3204 `MethodResponse` payload element in the HTTP response (see 7.10.2)

3205 Restrictions: None.

3206 B.3.4 DELETE instance resource

3207 This CIM-RS operation deletes an instance resource (see 7.6.2).

3208 This CIM-RS operation directly maps to the generic operation `DeleteInstance`.

3209 The input parameters for this generic operation are formed as follows:

- 3210 • the `InstancePath` parameter is formed from CIM namespace, class name and key bindings,
3211 which are all formed from information units extracted from the target URI of the HTTP request
3212 (see B.1.7)

3213 This generic operation has no output parameters.

3214 Restrictions: None..

3215 **B.3.5 GET instance resource**

3216 This CIM-RS operation retrieves an instance resource (see 7.6.3), possibly including associated or
3217 referenced instance resources.

3218 If neither the `$refer` nor the `$expand` query parameter is specified, this CIM-RS operation directly maps
3219 to the generic operation `GetInstance`.

3220 The input parameters for this generic operation are formed as follows:

- 3221 • the `InstancePath` parameter is formed from CIM namespace, class name and key bindings,
3222 which are all formed from information units extracted from the target URI of the HTTP request
3223 (see B.1.2)
- 3224 • the `IncludeClassOrigin` parameter is set to false
- 3225 • the `IncludedProperties` parameter is obtained from the `$properties` query parameter as
3226 described in Table

3227 The output parameters of this generic operation are used as follows:

- 3228 • the `Instance` parameter is used to form the `Instance` payload element in the HTTP
3229 response (see 7.6.1)

3230 **EXPERIMENTAL**

3231 If the `$refer` or `$expand` query parameters are specified, this CIM-RS operation maps to the generic
3232 operation `GetInstance` as described above, and possibly additional association traversal operations, as
3233 described in B.2.1.

3234 **EXPERIMENTAL**

3235 Restrictions:

- 3236 • Including the class origin of properties in the returned instance representation is not supported
3237 in CIM-RS.

3238 **B.3.6 GET page of instance collection resource**

3239 This CIM-RS operation retrieves the next page of a paged instance collection resource (see 7.8.2),
3240 resulting from enumeration by class, or from association traversal.

3241 This CIM-RS operation directly maps to the generic operation `PullInstancesWithPath`.

3242 The input parameters for this generic operation are formed as follows:

- 3243 • the `NamespacePath` parameter is formed from the CIM namespace, which is formed from
3244 information units extracted from the target URI of the HTTP request (see B.1.3 and B.1.4)
- 3245 • the `EnumerationContext` parameter is formed from the information about the next page to
3246 be retrieved within the overall collection, which is formed from information units extracted from
3247 the target URI of the HTTP request (see B.1.3 and B.1.4)
- 3248 • the `MaxObjectCount` parameter is obtained from the `$max` query parameter as described in
3249 Table

3250 The output parameters of this generic operation are used as follows:

- 3251 • the `InstanceList` parameter is used to form the `instances` attribute in the
3252 `InstanceCollection` payload element in the HTTP response (see 7.8.1)
- 3253 • if the `EndOfSequence` parameter is `FALSE`, the `EnumerationContext` parameter is used to
3254 form the information about the next page to be retrieved within the overall collection, in the URI
3255 for the `next` attribute in the `InstanceCollection` payload element in the HTTP response
3256 (see 7.8.1)
- 3257 • if the `EndOfSequence` parameter is `TRUE`, the `next` attribute is omitted from the
3258 `InstanceCollection` payload element in the HTTP response (see 7.8.1)

3259 Restrictions: None.

3260 B.3.7 GET page of reference collection resource

3261 This CIM-RS operation retrieves the next page of a paged reference collection resource (see 7.7.2),
3262 resulting from enumeration by class, or from association traversal.

3263 This CIM-RS operation directly maps to the generic operation `PullInstancePaths`.

3264 The input parameters for this generic operation are formed as follows:

- 3265 • the `NamespacePath` parameter is formed from the CIM namespace, which is formed from
3266 information units extracted from the target URI of the HTTP request (see B.1.3 and B.1.4)
- 3267 • the `EnumerationContext` parameter is formed from the information about the next page to
3268 be retrieved within the overall collection, which is formed from information units extracted from
3269 the target URI of the HTTP request (see B.1.3 and B.1.4)
- 3270 • the `MaxObjectCount` parameter is obtained from the `$max` query parameter as described in
3271 Table

3272 The output parameters of this generic operation are used as follows:

- 3273 • the `InstancePathList` parameter is used to form the `references` attribute in the
3274 `ReferenceCollection` payload element in the HTTP response (see 7.7.1)
- 3275 • if the `EndOfSequence` parameter is `FALSE`, the `EnumerationContext` parameter is used to
3276 form the information about the next page to be retrieved within the overall collection, in the URI
3277 for the `next` attribute in the `ReferenceCollection` payload element in the HTTP response
3278 (see 7.7.1)
- 3279 • if the `EndOfSequence` parameter is `TRUE`, the `next` attribute is omitted from the
3280 `ReferenceCollection` payload element in the HTTP response (see 7.7.1)

3281 Restrictions: None.

3282 B.3.8 GET instance enumeration resource

3283 This CIM-RS operation enumerates all instances of the specified class (including instances of subclasses)
3284 in the namespace of the targeted instance enumeration (see 7.9.1).

3285 If neither the `$refer` nor the `$expand` query parameter is specified, this CIM-RS operation directly maps
3286 to the generic operation `OpenEnumerateInstances`.

3287 The input parameters for this generic operation are formed as follows:

- 3288 • the `EnumClassPath` parameter is formed from:

- 3289 – the CIM namespace, formed from information units extracted from the target URI of the
3290 HTTP request (see B.1.5)
- 3291 – the class name, obtained from the `$class` query parameter in the target URI of the HTTP
3292 request (see B.1.5)
- 3293 • the `FilterQueryString` parameter is set from the `$filter` query parameter as described in
3294 Table
- 3295 • the `FilterQueryLanguage` parameter is set to "DMTF:FQL" (see C.2)
- 3296 • the `IncludeClassOrigin` parameter is set to false
- 3297 • the `IncludedProperties` parameter is set from the `$properties` query parameter as
3298 described in Table
- 3299 • the `ExcludeSubclassProperties` parameter is set to false
- 3300 • the `OperationTimeout` parameter is set from the `$pagingtimeout` query parameter as
3301 described in Table
- 3302 • the `ContinueOnError` parameter is set from the `$continueonerror` query parameter as
3303 described in Table
- 3304 • the `MaxObjectCount` parameter is set from the `$max` query parameter as described in Table

3305 The output parameters of this generic operation are used as follows:

- 3306 • the `InstanceList` parameter is used to form the `instances` attribute in the
3307 `InstanceCollection` payload element in the HTTP response (see 7.8.1)
- 3308 • if the `EndOfSequence` parameter is FALSE, the `EnumerationContext` parameter is used to
3309 form the information about the next page to be retrieved within the overall collection, in the URI
3310 for the `next` attribute in the `InstanceCollection` payload element in the HTTP response
3311 (see 7.8.1)
- 3312 • if the `EndOfSequence` parameter is TRUE, the `next` attribute is omitted from the
3313 `InstanceCollection` payload element in the HTTP response (see 7.8.1)

3314 EXPERIMENTAL

3315 If the `$refer` or `$expand` query parameters are specified, this CIM-RS operation maps to the generic
3316 operation `OpenEnumerateInstances` as described above, and possibly additional association traversal
3317 operations, as described in B.2.1.

3318 EXPERIMENTAL

3319 Restrictions:

- 3320 • Including the class origin of properties in the returned instance representations is not supported
3321 in CIM-RS.
- 3322 • Excluding subclass properties in the returned instance representations by setting a single
3323 indicator is not supported in CIM-RS (they can be excluded through the `$properties` query
3324 parameter).

3325 B.3.9 GET server entry point resource

3326 This CIM-RS operation retrieves the server entry point resource (see 7.12.2), which describes optional
3327 capabilities of the CIM-RS support, and information about the CIM namespaces of the server.

3328 This CIM-RS operation does not map to any generic operation.

3329 The CIM namespaces can be determined through the generic operation `GetInstance` on class
3330 `CIM_Namespace` in the Interop namespace. Alternatively, this information can be retrieved through direct
3331 interfaces.

3332 Restrictions: None.

3333 **B.3.10 PUT instance resource**

3334 This CIM-RS operation modifies some or all property values of an instance resource (see 7.6.4).

3335 This CIM-RS operation directly maps to the generic operation `ModifyInstance`.

3336 The input parameters for this generic operation are formed as follows:

- 3337 • the `InstancePath` parameter is formed from CIM namespace, class name and key bindings,
3338 which are all formed from information units extracted from the target URI of the HTTP request
3339 (see B.1.2)
- 3340 • the `ModifiedInstance` parameter is formed from the `instance` attribute of the `Instance`
3341 payload element in the HTTP request (see 7.6.1)
- 3342 • the `IncludedProperties` parameter is obtained from the `$properties` query parameter as
3343 described in Table

3344 This generic operation does not have any output parameters.

3345 Restrictions: None.

3346 **B.4 Listener operations**

3347 This subclause describes a listener's decision tree for how incoming CIM-RS listener operations are to be
3348 analyzed, identified, and mapped to generic listener operations: For each HTTP method, the listener will
3349 examine its target URI. Based upon the listener's defined URI structure, it will determine what type of
3350 resource is targeted, and will then determine which generic operations are to be invoked.

3351 The following subclauses describe each combination of HTTP method and resource type.

3352 **B.4.1 POST listener destination resource**

3353 This CIM-RS listener operation delivers an indication to a listener (see 7.11.2).

3354 This CIM-RS operation directly maps to the generic operation `DeliverIndication`.

3355 The input parameters for this generic operation are formed as follows:

- 3356 • the `ListenerDestination` parameter is formed from information units extracted from the
3357 target URI of the HTTP request (see B.1.8)
- 3358 • the `Indication` parameter is formed from the `indication` attribute of the
3359 `IndicationDeliveryRequest` payload element in the HTTP request (see 7.11.1)

3360 This generic operation does not have any output parameters.

3361 Restrictions: None.

3362 B.4.2 GET listener entry point resource

3363 This CIM-RS operation retrieves the listener entry point resource (see 7.13.2), which describes optional
3364 capabilities of the CIM-RS support.

3365 This CIM-RS operation does not map to any generic operation.

3366 Restrictions: None.

ANNEX C (informative)

Mapping generic operations to CIM-RS

3371 This annex describes how generic operations (see [DSP0223](#)) are to be mapped to CIM-RS operations,
3372 resources, and query parameters. This mapping is provided primarily to describe how the CIM-RS
3373 protocol conforms to generic operations. This mapping can also be used to translate operation
3374 requirements defined in management profiles that are stated in terms of generic operations, into CIM-RS
3375 operations. The latter may be useful for implementations of CIM servers that define their provider API in
3376 terms of CIM-RS operations.

3377 C.1 Conformance

3378 CIM-RS does not satisfy all conformance requirements defined in generic operations ([DSP0223](#)). As a
3379 result, CIM-RS is not a conforming WBEM protocol. The subclauses in this annex provide details.

3380 C.2 Support of optional generic operations features

3381 This subclause describes how CIM-RS supports optional features defined in generic operations.

- 3382 • CIM-RS does not support client side control of returning class origin information (generic
3383 operation parameter `IncludeClassOrigin`)
- 3384 • CIM-RS supports error handling by means of returning DMTF standard messages (also known
3385 as "extended error handling")
- 3386 • CIM-RS supports filter queries in pulled instance enumeration operations. However, only the
3387 upcoming DMTF *Filter Query Language* will be supported. In anticipation of that, the
3388 `FilterQueryLanguage` parameter of any generic operations is set to "DMTF:FQL"..
- 3389 • CIM-RS supports client side control of continuation on error for pulled instance enumeration
3390 operations

3391 C.3 Operations supported

3392 This subclause describes generic operations that are supported in CIM-RS.

3393 C.3.1 GetInstance

3394 This generic operation is supported via HTTP GET on an instance resource (see 7.6.3).

3395 Its input parameters map to CIM-RS as follows:

- 3396 • `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3397 • `IncludeClassOrigin`: Not supported in CIM-RS (optional in [DSP0223](#))
- 3398 • `IncludedProperties`: `$properties` query parameter (see Table B-1)

3399 Its output parameters map to CIM-RS as follows:

- 3400 • `Instance`: `Instance` payload element in HTTP response (see 7.6.1)

3401 Conformance: Yes.

3402 C.3.2 DeleteInstance

3403 This generic operation is supported via HTTP DELETE on an instance resource (see 7.6.2).

3404 Its input parameters map to CIM-RS as follows:

- 3405 • `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)

3406 This generic operation has no output parameters.

3407 Conformance: Yes.

3408 C.3.3 ModifyInstance

3409 This generic operation is supported via HTTP PUT on an instance resource (see 7.6.4).

3410 Its input parameters map to CIM-RS as follows:

- 3411 • `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3412 • `ModifiedInstance`: Instance payload element in HTTP request (see 7.6.1)
- 3413 • `IncludedProperties`: `$properties` query parameter (see Table B-1)

3414 This generic operation has no output parameters.

3415 Conformance: Yes.

3416 C.3.4 CreateInstance

3417 This generic operation is supported via HTTP POST on an instance creation resource (see 7.5.1).

3418 Its input parameters map to CIM-RS as follows:

- 3419 • `ClassPath`: Information units in target URI of the HTTP request (see B.1.1)
- 3420 • `NewInstance`: Instance payload element in HTTP request (see 7.6.1)

3421 Its output parameters map to CIM-RS as follows:

- 3422 • `InstancePath`: Location header field in HTTP response (see 7.5.1)

3423 Conformance: Yes.

3424 C.3.5 OpenEnumerateInstances

3425 This generic operation is supported via HTTP GET on an instance enumeration resource (see 7.9.1).

3426 Its input parameters map to CIM-RS as follows:

- 3427 • `EnumClassPath`: Information units in target URI of the HTTP request (see B.1.5)
- 3428 • `FilterQueryString`: `$filter` query parameter (see Table B-1)
- 3429 • `FilterQueryLanguage`: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- 3430 • `IncludeClassOrigin`: Not supported in CIM-RS (optional in [DSP0223](#))
- 3431 • `IncludedProperties`: `$properties` query parameter (see Table B-1)
- 3432 • `ExcludeSubclassProperties`: Not supported directly; can be achieved with `$properties`
- 3433 • `query parameter` (see Table B-1)

- 3434
- `OperationTimeout: $pagingtimeout` query parameter (see Table B-1)
- 3435
- `ContinueOnError: $continueonerror` query parameter (see Table B-1)
- 3436
- `MaxObjectCount: $max` query parameter (see Table B-1)

3437 Its output parameters map to CIM-RS as follows:

- 3438
- `InstanceList: instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3439
- 3440
- `EnumerationContext: information units in URI of next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3441
- 3442
- `EndOfSequence: omission or presence of next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3443

3444 Conformance: Yes.

3445 C.3.6 OpenEnumerateInstancePaths

3446 This generic operation is supported via HTTP GET on an instance enumeration resource (see 7.9.1),
3447 where its `$properties` query parameter is set to include no properties.

3448 Its input parameters map to CIM-RS as follows:

- 3449
- `EnumClassPath: Information units in target URI of the HTTP request` (see B.1.5)
- 3450
- `FilterQueryString: $filter` query parameter (see Table B-1)
- 3451
- `FilterQueryLanguage: Only "DMTF:FQL"` is supported by CIM-RS (see C.2)
- 3452
- `OperationTimeout: $pagingtimeout` query parameter (see Table B-1)
- 3453
- `ContinueOnError: $continueonerror` query parameter (see Table B-1)
- 3454
- `MaxObjectCount: $max` query parameter (see Table B-1)

3455 Its output parameters map to CIM-RS as follows:

- 3456
- `InstancePathList: instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3457
- 3458
- `EnumerationContext: information units in URI of next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3459
- 3460
- `EndOfSequence: omission or presence of next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- 3461

3462 Conformance: Yes.

3463 C.3.7 OpenAssociators (EXPERIMENTAL)

3464 EXPERIMENTAL

3465 This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a
3466 `$properties` query parameter that specifies not to include any properties, and with a `$expand` query
3467 parameter that specifies each association to be traversed (for example,
3468 `$expand=AssociationClassName.[AssociatedClassName]AssociatedRoleName`).

3469 Its input parameters map to CIM-RS as follows:

- 3470 • `SourceInstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3471 • `AssociationClassName`: association class in `$expand` query parameter (see B.2.1)
- 3472 • `AssociatedClassName`: associated class filter in `$expand` query parameter (see B.2.1)
- 3473 • `SourceRoleName`: Not supported in CIM-RS (mandatory in [DSP0223](#))
- 3474 • `AssociatedRoleName`: association end in `$expand` query parameter (see Table B-1)
- 3475 • `FilterQueryString`: `$filter` query parameter (see Table B-1)
- 3476 • `FilterQueryLanguage`: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- 3477 • `IncludeClassOrigin`: Not supported in CIM-RS (optional in [DSP0223](#))
- 3478 • `IncludedProperties`: `$properties` query parameter (see Table B-1) specifying properties
- 3479 in the navigation properties included via the `$expand` query parameter
- 3480 • `ExcludeSubclassProperties`: Not supported directly; can be achieved with the
- 3481 `$properties` query parameter (see Table B-1) specifying properties in the navigation
- 3482 properties included via the `$expand` query parameter
- 3483 • `OperationTimeout`: `$pagingtimeout` query parameter (see Table B-1)
- 3484 • `ContinueOnError`: `$continueonerror` query parameter (see Table B-1)
- 3485 • `MaxObjectCount`: `$max` query parameter (see Table B-1)

3486 Its output parameters map to CIM-RS as follows:

- 3487 • `InstanceList`: `instances` attribute of `InstanceCollection` payload element in HTTP
- 3488 response (see 7.8.1)
- 3489 • `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection`
- 3490 payload element in HTTP response (see 7.8.1)
- 3491 • `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload
- 3492 element in HTTP response (see 7.8.1)

3493 Conformance: No, for the following reasons:

- 3494 • the mandatory `SourceRoleName` filter is not supported
- 3495 • traversal of all referencing associations without knowing them upfront is not supported

3496 **EXPERIMENTAL**

3497 **C.3.8 OpenAssociatorPaths (EXPERIMENTAL)**

3498 **EXPERIMENTAL**

3499 This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a
 3500 `$properties` query parameter that specifies not to include any properties, and with a `$refer` query
 3501 parameter that specifies each association to be traversed (for example,
 3502 `$refer=AssociationClassName.[AssociatedClassName]AssociatedRoleName`).

3503 Its input parameters map to CIM-RS as follows:

- 3504 • `SourceInstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3505 • `AssociationClassName`: association class in `$refer` query parameter (see B.2.1)

- 3506 • `AssociatedClassName`: associated class filter in `$refer` query parameter (see B.2.1)
- 3507 • `SourceRoleName`: Not supported in CIM-RS (mandatory in [DSP0223](#))
- 3508 • `AssociatedRoleName`: association end in `$refer` query parameter (see B.2.1)
- 3509 • `FilterQueryString`: `$filter` query parameter (see Table B-1)
- 3510 • `FilterQueryLanguage`: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- 3511 • `IncludeClassOrigin`: Not supported in CIM-RS (optional in [DSP0223](#))
- 3512 • `IncludedProperties`: `$properties` query parameter (see Table B-1) specifying properties
- 3513 in the navigation properties included via the `$refer` query parameter
- 3514 • `ExcludeSubclassProperties`: Not supported directly; can be achieved with the
- 3515 `$properties` query parameter (see Table B-1) specifying properties in the navigation
- 3516 properties included via the `$refer` query parameter
- 3517 • `OperationTimeout`: `$pagingtimeout` query parameter (see Table B-1)
- 3518 • `ContinueOnError`: `$continueonerror` query parameter (see Table B-1)
- 3519 • `MaxObjectCount`: `$max` query parameter (see Table B-1)

3520 Its output parameters map to CIM-RS as follows:

- 3521 • `InstancePathList`: `instances` attribute of `InstanceCollection` payload element in
- 3522 HTTP response (see 7.8.1)
- 3523 • `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection`
- 3524 payload element in HTTP response (see 7.8.1)
- 3525 • `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload
- 3526 element in HTTP response (see 7.8.1)

3527 Conformance: No, for the following reasons:

- 3528 • the mandatory `SourceRoleName` filter is not supported
- 3529 • traversal of all referencing associations without knowing them upfront is not supported

3530 EXPERIMENTAL

3531 C.3.9 OpenReferences (EXPERIMENTAL)

3532 EXPERIMENTAL

3533 This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a

3534 `$properties` query parameter that specifies not to include any properties, and with a `$expand` query

3535 parameter that specifies each association to be returned (for example,

3536 `$expand=AssociationClassName`).

3537 Its input parameters map to CIM-RS as follows:

- 3538 • `SourceInstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3539 • `AssociationClassName`: association class in `$expand` query parameter (see B.2.1)
- 3540 • `AssociatedClassName`: associated class filter in `$expand` query parameter (see B.2.1)

- 3541 • `SourceRoleName`: Not supported in CIM-RS (mandatory in [DSP0223](#))
- 3542 • `AssociatedRoleName`: association end in `$expand` query parameter (see B.2.1)
- 3543 • `FilterQueryString`: `$filter` query parameter (see Table B-1)
- 3544 • `FilterQueryLanguage`: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- 3545 • `IncludeClassOrigin`: Not supported in CIM-RS (optional in [DSP0223](#))
- 3546 • `IncludedProperties`: `$properties` query parameter (see Table B-1) specifying properties
3547 in the navigation properties included via the `$expand` query parameter
- 3548 • `ExcludeSubclassProperties`: Not supported directly; can be achieved with the
3549 `$properties` query parameter (see Table B-1) specifying properties in the navigation
3550 properties included via the `$expand` query parameter
- 3551 • `OperationTimeout`: `$pagingtimeout` query parameter (see Table B-1)
- 3552 • `ContinueOnError`: `$continueonerror` query parameter (see Table B-1)
- 3553 • `MaxObjectCount`: `$max` query parameter (see Table B-1)

3554 Its output parameters map to CIM-RS as follows:

- 3555 • `InstanceList`: `instances` attribute of `InstanceCollection` payload element in HTTP
3556 response (see 7.8.1)
- 3557 • `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection`
3558 payload element in HTTP response (see 7.8.1)
- 3559 • `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload
3560 element in HTTP response (see 7.8.1)

3561 Conformance: No, for the following reasons:

- 3562 • the mandatory `SourceRoleName` filter is not supported
- 3563 • return of all referencing associations without knowing them upfront is not supported

3564 EXPERIMENTAL

3565 C.3.10 OpenReferencePaths (EXPERIMENTAL)

3566 EXPERIMENTAL

3567 This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a
3568 `$properties` query parameter that specifies not to include any properties, and with a `$refer` query
3569 parameter that specifies each association to be returned (for example,
3570 `$refer=AssociationClassName`).

3571 Its input parameters map to CIM-RS as follows:

- 3572 • `SourceInstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3573 • `AssociationClassName`: association class in `$refer` query parameter (see B.2.1)
- 3574 • `AssociatedClassName`: associated class filter in `$refer` query parameter (see B.2.1)
- 3575 • `SourceRoleName`: Not supported in CIM-RS (mandatory in [DSP0223](#))
- 3576 • `AssociatedRoleName`: association end in `$refer` query parameter (see B.2.1)

- 3577 • `FilterQueryString`: `$filter` query parameter (see Table B-1)
- 3578 • `FilterQueryLanguage`: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- 3579 • `IncludeClassOrigin`: Not supported in CIM-RS (optional in [DSP0223](#))
- 3580 • `IncludedProperties`: `$properties` query parameter (see Table B-1) specifying properties
3581 in the navigation properties included via the `$refer` query parameter
- 3582 • `ExcludeSubclassProperties`: Not supported directly; can be achieved with the
3583 `$properties` query parameter (see Table B-1) specifying properties in the navigation
3584 properties included via the `$refer` query parameter
- 3585 • `OperationTimeout`: `$pagingtimeout` query parameter (see Table B-1)
- 3586 • `ContinueOnError`: `$continueonerror` query parameter (see Table B-1)
- 3587 • `MaxObjectCount`: `$max` query parameter (see Table B-1)

3588 Its output parameters map to CIM-RS as follows:

- 3589 • `InstancePathList`: `instances` attribute of `InstanceCollection` payload element in
3590 HTTP response (see 7.8.1)
- 3591 • `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection`
3592 payload element in HTTP response (see 7.8.1)
- 3593 • `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload
3594 element in HTTP response (see 7.8.1)

3595 Conformance: No, for the following reasons:

- 3596 • the mandatory `SourceRoleName` filter is not supported
- 3597 • return of all referencing associations without knowing them upfront is not supported

3598 EXPERIMENTAL

3599 C.3.11 PullInstancesWithPath

3600 This generic operation is supported via HTTP GET on a page of an instance collection resource (see
3601 7.8.2), that had been created (via the `$properties` query parameter) such that properties were to be
3602 returned.

3603 Its input parameters map to CIM-RS as follows:

- 3604 • `NamespacePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3605 • `EnumerationContext`: information units in target URI of the HTTP request (see B.1.2)
- 3606 • `MaxObjectCount`: `$max` query parameter (see Table B-1)

3607 Its output parameters map to CIM-RS as follows:

- 3608 • `InstanceList`: `instances` attribute of `InstanceCollection` payload element in HTTP
3609 response (see 7.8.1)
- 3610 • `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection`
3611 payload element in HTTP response (see 7.8.1)
- 3612 • `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload
3613 element in HTTP response (see 7.8.1)

3614 Conformance: Yes.

3615 C.3.12 PullInstancePaths

3616 This generic operation is supported via HTTP GET on a page of an instance collection resource (see
3617 7.8.2), that had been created (via the `$properties` query parameter) such that no properties were to be
3618 returned.

3619 Its input parameters map to CIM-RS as follows:

- 3620 • `NamespacePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3621 • `EnumerationContext`: information units in target URI of the HTTP request (see B.1.2)
- 3622 • `MaxObjectCount`: `$max` query parameter (see Table B-1)

3623 Its output parameters map to CIM-RS as follows:

- 3624 • `InstanceList`: `instances` attribute of `InstanceCollection` payload element in HTTP
3625 response (see 7.8.1)
- 3626 • `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection`
3627 payload element in HTTP response (see 7.8.1)
- 3628 • `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload
3629 element in HTTP response (see 7.8.1)

3630 Conformance: Yes.

3631 C.3.13 InvokeMethod

3632 This generic operation is supported via HTTP POST on a non-static method invocation resource (see
3633 7.10.3).

3634 Its input parameters map to CIM-RS as follows:

- 3635 • `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- 3636 • `MethodName`: `method` attribute of `MethodRequest` payload element in HTTP request (see
3637 7.10.1)
- 3638 • `InParmValues`: `parameters` attribute of `MethodRequest` payload element in HTTP request
3639 (see 7.10.1)

3640 Its output parameters map to CIM-RS as follows:

- 3641 • `OutParmValues`: `parameters` attribute of `MethodResponse` payload element in HTTP
3642 response (see 7.10.2)
- 3643 • `ReturnValue`: `returnvalue` attribute of `MethodResponse` payload element in HTTP
3644 response (see 7.10.2)

3645 Conformance: Yes.

3646 C.3.14 InvokeStaticMethod

3647 This generic operation is supported via HTTP POST on a static method invocation resource (see 7.10.3).

3648 Its input parameters map to CIM-RS as follows:

- 3649 • `ClassPath`: Information units in target URI of the HTTP request (see B.1.2)

- 3650 • MethodName: method attribute of MethodRequest payload element in HTTP request (see
3651 7.10.1)
- 3652 • InParmValues: parameters attribute of MethodRequest payload element in HTTP request
3653 (see 7.10.1)

3654 Its output parameters map to CIM-RS as follows:

- 3655 • OutParmValues: parameters attribute of MethodResponse payload element in HTTP
3656 response (see 7.10.2)
- 3657 • ReturnValue: returnvalue attribute of MethodResponse payload element in HTTP
3658 response (see 7.10.2)

3659 Conformance: Yes.

3660 C.4 Operations not supported

3661 The following generic operations are not supported in CIM-RS.

3662 C.4.1 Direct instance enumeration operations

3663 Direct instance enumeration operations are not supported in CIM-RS, because it is always possible that
3664 the resulting collections in CIM-RS are paged.

3665 **Table C-1 – Pulled equivalents of direct instance enumeration operations**

Unsupported Direct Enumeration Operation	Supported Pulled Equivalent
EnumerateInstances	OpenEnumerateInstances (Section C.3.5)
EnumerateInstanceNames	OpenEnumerateInstancePaths (Section C.3.6)
Associators	OpenAssociators (Section C.3.7)
AssociatorNames	OpenAssociatorPaths (Section C.3.8)
References	OpenReferences (Section C.3.9)
GetReferencingInstancesPaths	OpenReferencePaths (Section C.3.10)

3666

3667 C.4.2 Class and qualifier type operations

3668 Class and qualifier type operations are not supported in CIM-RS.

- 3669 • GetClass
- 3670 • DeleteClass
- 3671 • ModifyClass
- 3672 • CreateClass
- 3673 • EnumerateClasses
- 3674 • EnumerateClassNames

- 3675 • AssociatorClasses
- 3676 • AssociatorClassPaths
- 3677 • ReferenceClasses
- 3678 • ReferenceClassPaths
- 3679 • GetQualifierType
- 3680 • DeleteQualifierType
- 3681 • CreateQualifierType
- 3682 • EnumerateQualifierTypes

3683 **C.4.3 Other operations**

3684 The following other generic operations are not supported in CIM-RS.

- 3685 • OpenQueryInstances
- 3686 • PullInstances
- 3687 • EnumerationCount
- 3688 • CloseEnumeration

ANNEX D
(informative)

Examples

D.1 Navigation between resources (EXPERIMENTAL)

EXPERIMENTAL

This annex provides examples on how to navigate between resources using the \$expand (see 6.5.3) and \$refer (see 6.5.9) query parameters. For a description of the concepts for navigating between resources, see 5.6.

D.1.1 Classes and instances used in the examples

The examples use the classes from the class diagram shown in Figure D-1.

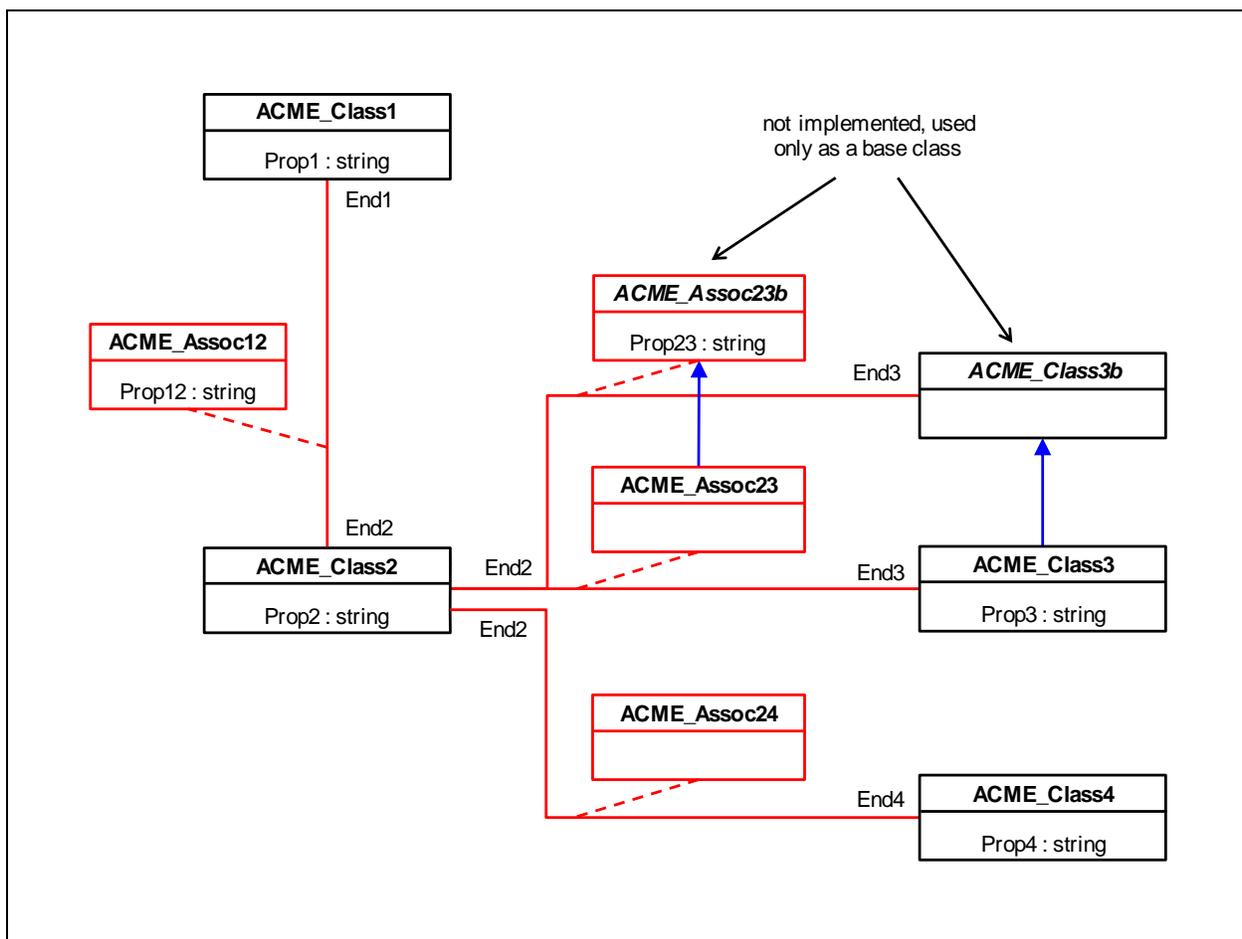


Figure D-1 – Class diagram for navigation examples

The representations of results uses an informal notation that indicates nesting of elements by indentation.

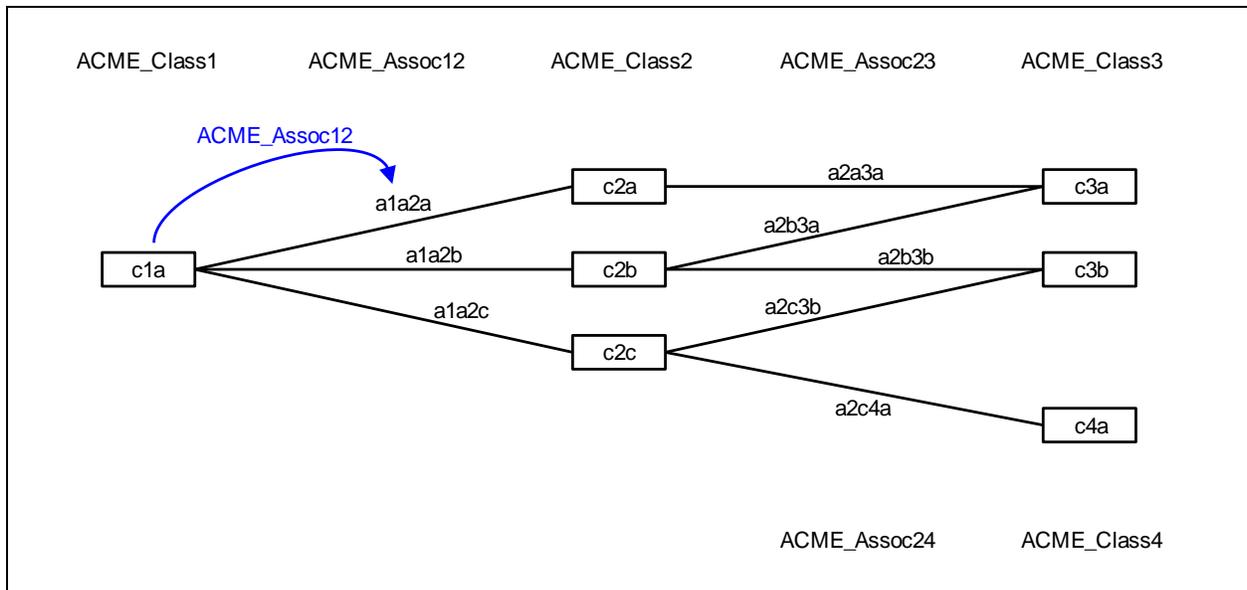
3704 The examples are limited to requests for instance retrieval, for brevity. Requests for retrieval of instance
3705 collections work the same way, except that each instance in the collection is affected.

3706 The following MOF defines the classes shown in Figure D-1:

```
3707 class ACME_Class1 { string Prop1; };
3708
3709 class ACME_Class2 { string Prop2; };
3710
3711 [Abstract]
3712 class ACME_Class3b { }; // not implemented
3713
3714 class ACME_Class3 : ACME_Class3b { string Prop3; };
3715
3716 [Association]
3717 class ACME_Assoc12 {
3718     ACME_Class1 REF End1;
3719     ACME_Class2 REF End2;
3720     string Prop12;
3721 };
3722
3723 [Association, Abstract]
3724 class ACME_Assoc23b { // not implemented
3725     ACME_Class2 REF End2;
3726     ACME_Class3b REF End3;
3727     string Prop23;
3728 };
3729
3730 [Association]
3731 class ACME_Assoc23 : ACME_Assoc23b {
3732     [Override("End3")] ACME_Class3 REF End3; // now references the subclass
3733 };
3734
3735 [Association]
3736 class ACME_Assoc24 {
3737     ACME_Class2 REF End2;
3738     ACME_Class4 REF End4;
3739 };
```

3740 **D.1.2 Navigation to referencing association instances**

3741 In this example, the client retrieves an instance and specifies a navigation path that identifies association
 3742 instances that reference the instance being retrieved. Figure D-2 shows the instance diagram and the
 3743 blue navigation path "ACME_Assoc12", starting at instance c1a.



3744
3745

3746 **Figure D-2 – Example instance diagram for navigation to referencing association instances**

3747 An instance retrieval request using this navigation path with the \$refer query parameter will return the
 3748 following instance representation:

```
3749 GET /c1a?$refer=ACME_Assoc12
3750
3751 Instance c1a:
3752   Prop1: "..."/>

```

3757 An instance retrieval request using this navigation path with the \$expand query parameter will return the
 3758 following instance representation:

```
3759 GET /c1a?$expand=ACME_Assoc12
3760
3761 Instance c1a:
3762   Prop1: "..."/>

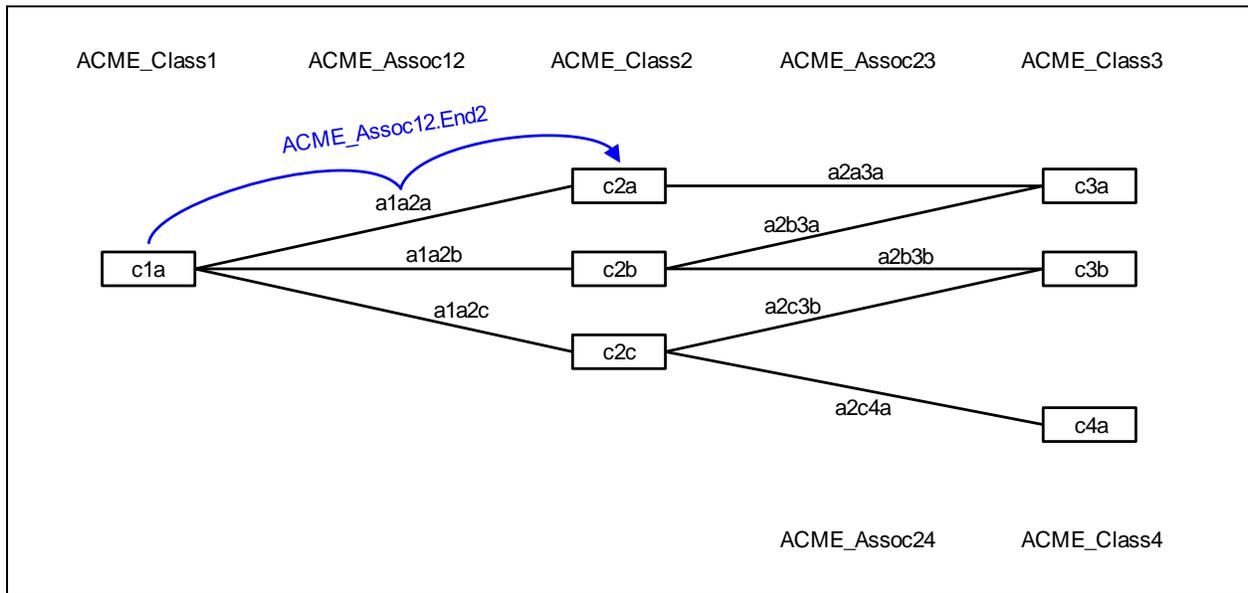
```

```

3769     End1: ref c1a
3770     End2: ref c2b
3771     Prop12: "...
3772 Instance a1a2c:
3773     End1: ref c1a
3774     End2: ref c2c
3775     Prop12: "...
    
```

3776 **D.1.3 Navigation to associated instances**

3777 In this example, the client retrieves an instance and specifies a navigation path that identifies the
 3778 instances associated to the instance being retrieved. Figure D-3 shows the instance diagram and the blue
 3779 navigation path "ACME_Assoc12.End2", starting at instance c1a.



3780
3781

3782 **Figure D-3 – Example instance diagram for navigation to associated instances**

3783 An instance retrieval request using this navigation path with the \$refer query parameter will return the
 3784 following instance representation:

```

3785 GET /c1a?$refer=ACME_Assoc12.End2
3786
3787 Instance c1a:
3788     Prop1: "...
3789     ACME_Assoc12.End2: ReferenceCollection:
3790         ref c2a
3791         ref c2b
3792         ref c2c
    
```

3793 An instance retrieval request using this navigation path with the \$expand query parameter will return the
 3794 following instance representation:

```

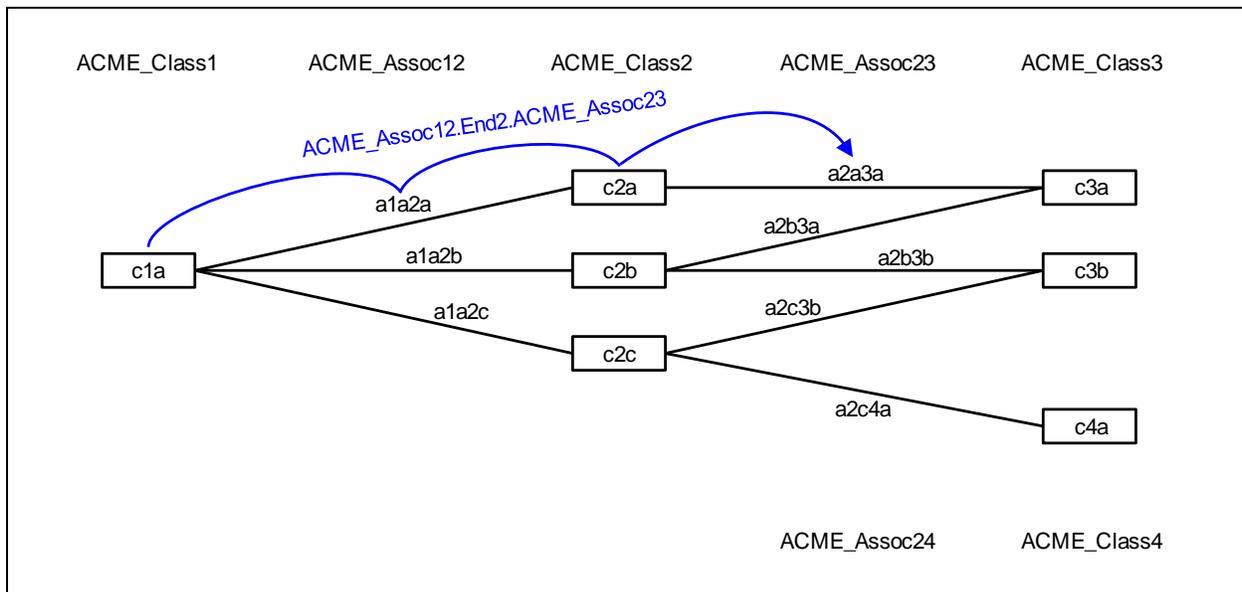
3795 GET /c1a?$expand=ACME_Assoc12.End2
3796
    
```

```

3797 Instance c1a:
3798   Prop1: "...
3799   ACME_Assoc12.End2: InstanceCollection:
3800     Instance c2a:
3801       Prop2: "...
3802     Instance c2b:
3803       Prop2: "...
3804     Instance c2c:
3805       Prop2: "...
    
```

3806 **D.1.4 Navigation to association instances across one hop**

3807 In this example, the client retrieves an instance and specifies a navigation path that identifies the
 3808 association instances that reference the instances associated to the instance being retrieved. Figure D-4
 3809 shows the instance diagram and the blue navigation path "ACME_Assoc12.End2.ACME_Assoc23",
 3810 starting at instance c1a.



3811
3812

3813 **Figure D-4 – Example instance diagram for navigation to association instances across one hop**

3814 An instance retrieval request using this navigation path with the \$refer query parameter will return the
 3815 following instance representation:

```

3816 GET /c1a?$refer=ACME_Assoc12.End2.ACME_Assoc23
3817
3818 Instance c1a:
3819   Prop1: "...
3820   ACME_Assoc12.End2.ACME_Assoc23: ReferenceCollection:
3821     ref a2a3a
3822     ref a2b3a
3823     ref a2b3b
3824     ref a2c3b
    
```

3825 Note that instances of association class ACME_Assoc24 are not included, because navigation across
3826 ACME_Assoc23 was requested.

3827 An instance retrieval request using this navigation path with the \$expand query parameter will return the
3828 following instance representation:

```
3829 GET /c1a?$expand=ACME_Assoc12.End2.ACME_Assoc23
```

3830

```
3831 Instance c1a:
```

```
3832   Prop1: "..."
```

```
3833   ACME_Assoc12.End2.ACME_Assoc23: InstanceCollection:
```

```
3834     Instance a2a3a:
```

```
3835       End2: ref c2a
```

```
3836       End3: ref c3a
```

```
3837       Prop23: "..."
```

```
3838     Instance a2b3a:
```

```
3839       End2: ref c2b
```

```
3840       End3: ref c3a
```

```
3841       Prop23: "..."
```

```
3842     Instance a2b3b:
```

```
3843       End2: ref c2b
```

```
3844       End3: ref c3b
```

```
3845       Prop23: "..."
```

```
3846     Instance a2c3b:
```

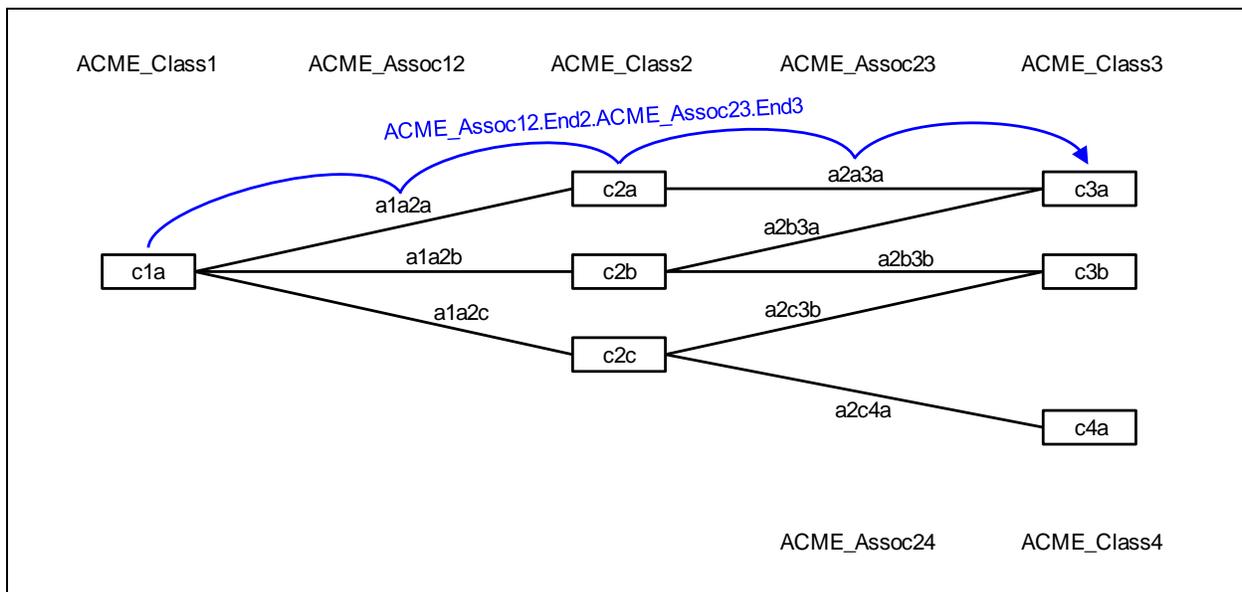
```
3847       End2: ref c2c
```

```
3848       End3: ref c3b
```

```
3849       Prop23: "..."
```

3850 **D.1.5 Navigation to associated instances across two hops**

3851 In this example, the client retrieves an instance and specifies a navigation path that identifies instances
 3852 associated to the instance being retrieved across two specific association hops. Figure D-5 shows the
 3853 instance diagram and the blue navigation path "ACME_Assoc12.End2.ACME_Assoc23.End3", starting at
 3854 instance c1a.



3855
3856

3857 **Figure D-5 – Example instance diagram for navigation to associated instances across two hops**

3858 An instance retrieval request using this navigation path with the \$refer query parameter will return the
 3859 following instance representation:

```
3860 GET /c1a?$refer=ACME_Assoc12.End2.ACME_Assoc23.End3
3861
3862 Instance c1a:
3863   Prop1: "...
3864   ACME_Assoc12.End2.ACME_Assoc23.End3: ReferenceCollection:
3865     ref c3a
3866     ref c3a
3867     ref c3b
3868     ref c3b
```

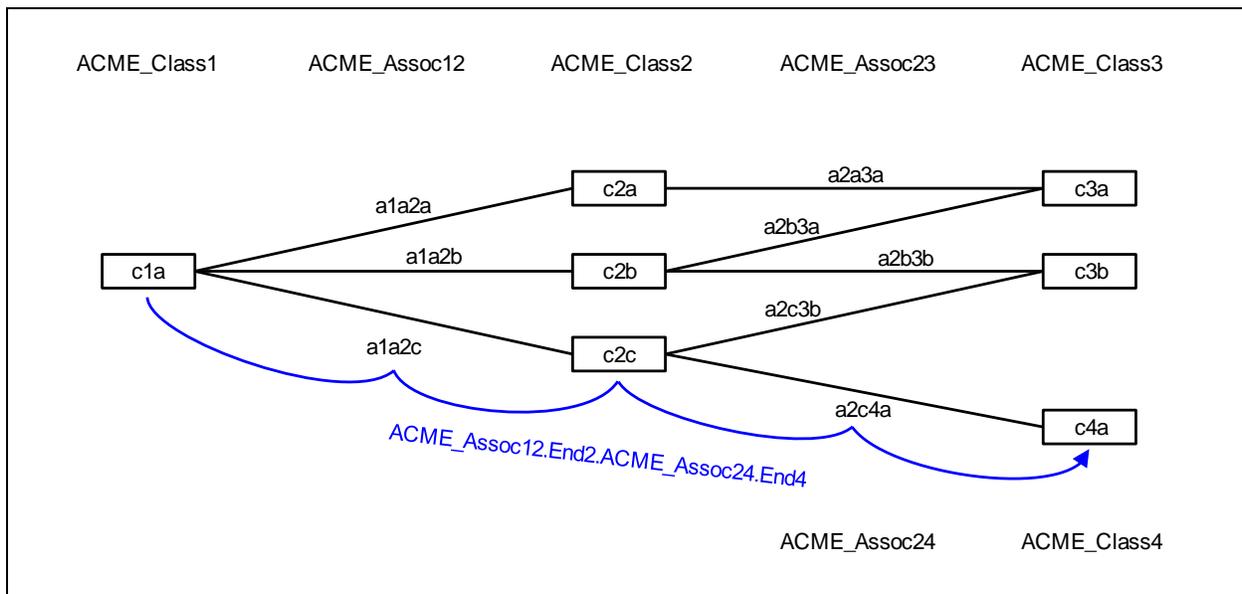
3869 Note that instances c3a and c3b each occur two times in the list. The reason for this is that the inclusion
 3870 is driven strictly by the navigation paths that lead to the desired target, and there is no optimization to
 3871 reduce any duplicates.

3872 Note that instances of class ACME_Class4 are not included, because navigation across ACME_Assoc23
 3873 and its End3 was requested.

3874 An instance retrieval request using this navigation path with the \$expand query parameter will also return
 3875 the same duplicates and is not shown, for brevity.

3876 **D.1.6 Navigation to associated instances across two hops (2)**

3877 This example is similar to the previous example, except that the navigation path uses the other possible
 3878 association for the second hop. Figure D-6 shows the instance diagram and the blue navigation path
 3879 "ACME_Assoc12.End2.ACME_Assoc24.End4", starting at instance c1a.



3880
 3881
 3882 **Figure D-6 – Example instance diagram for navigation to associated instances across two hops (2)**

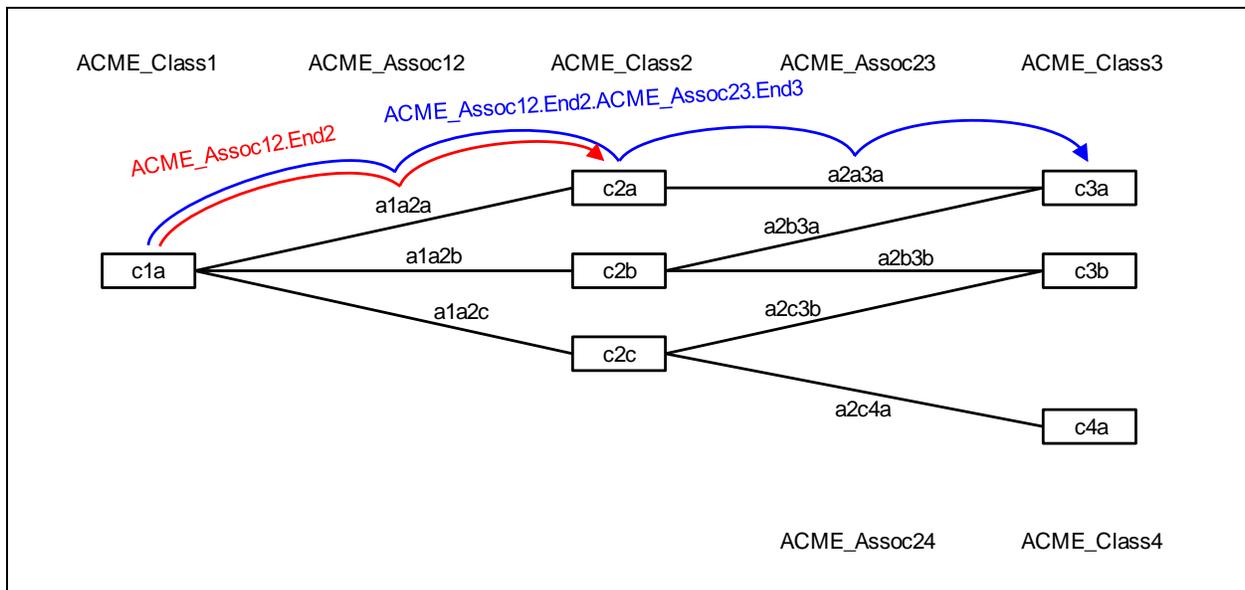
3883 An instance retrieval request using this navigation path with the \$refer query parameter will return the
 3884 following instance representation:

```
3885 GET /c1a?$refer=ACME_Assoc12.End2.ACME_Assoc24.End4
3886
3887 Instance c1a:
3888     Prop1: "...
3889     ACME_Assoc12.End2.ACME_Assoc24.End4: ReferenceCollection:
3890     ref c4a
```

3891 Note that the intermediate instances of class ACME_Class2 do not show up in the result. Some of them
 3892 are being traversed in the course of getting to the result instances, but because only the end result is
 3893 represented, the navigation path to get there does not show up.

3894 D.1.7 Navigation with two paths that form a subset (merge)

3895 In this example, the client retrieves an instance and specifies two navigation path: one that identifies
 3896 instances directly associated to the instance being retrieved, and one that identifies instances associated
 3897 across one additional association hop. Figure D-7 shows the instance diagram and the two navigation
 3898 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
 3899 is used with \$expand.



3900
3901

3902 **Figure D-7 – Example instance diagram for navigation with two paths that form a subset (merge)**

3903 An instance retrieval request using these two navigation paths with the \$refer query parameter will
 3904 return the following instance representation:

```
3905 GET /c1a?$refer=ACME_Assoc12.End2,ACME_Assoc12.End2.ACME_Assoc23.End3
```

3906

```
3907 Instance c1a:
```

```
3908   Prop1: "..."
```

```
3909   ACME_Assoc12.End2: ReferenceCollection:
```

```
3910     ref c2a
```

```
3911     ref c2b
```

```
3912     ref c2c
```

```
3913   ACME_Assoc12.End2.ACME_Assoc23.End3: ReferenceCollection:
```

```
3914     ref c3a
```

```
3915     ref c3a
```

```
3916     ref c3b
```

```
3917     ref c3b
```

3918 Note that the two navigation properties have not been merged, even though one navigation path was a
3919 subset of the other. The reason is that the shorter one was not expanded to instances.

3920 A changed request where the shorter navigation path is used with the `$expand` query parameter and the
3921 longer one is used with `$refer` will return the following instance representation:

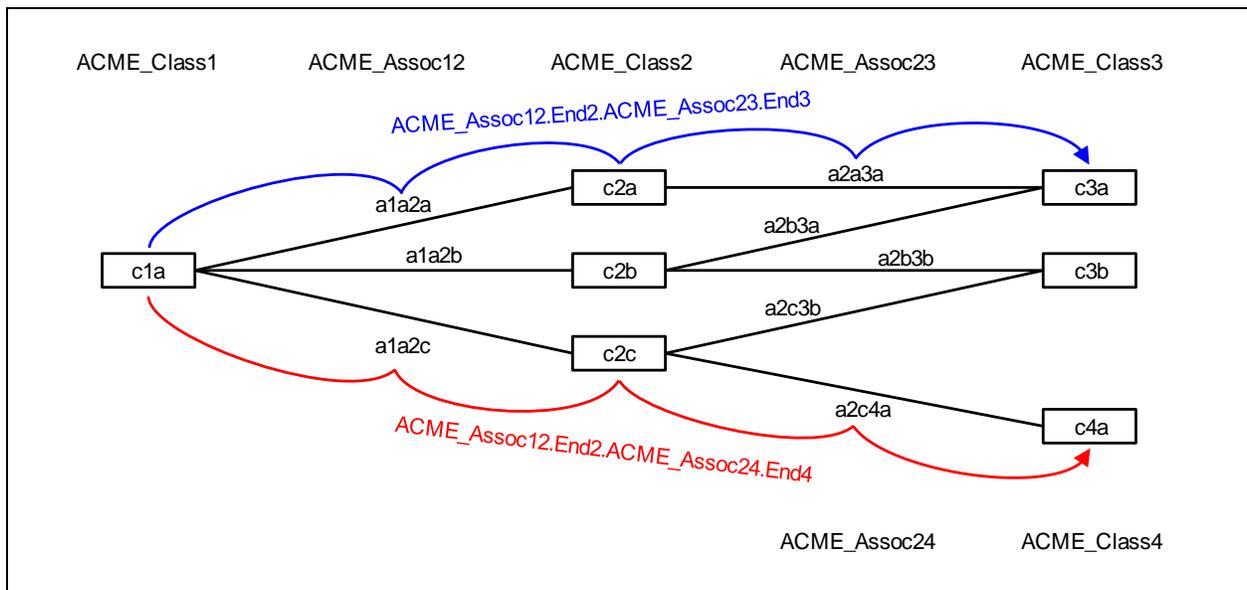
```
3922 GET /c1a?$expand=ACME_Assoc12.End2&$refer=ACME_Assoc12.End2.ACME_Assoc23.End3
3923
3924 Instance c1a:
3925     Prop1: "..."/>

```

3940 Note that the two navigation properties now have been merged, and that the names of the inner
 3941 navigation properties are relative to their starting point (that is, just "ACME_Assoc23.End3" and not
 3942 "ACME_Assoc12.End2.ACME_Assoc23.End3" as specified in the query parameter).

3943 **D.1.8 Navigation with two paths that have a common begin**

3944 This example is similar to the previous one, except that the two navigation paths have a common path
 3945 after their start but none is a subset of the other. Figure D-8 shows the instance diagram and the two
 3946 navigation paths, in blue and red.



3947
 3948

3949 **Figure D-8 – Example instance diagram for navigation with two paths that have a common begin**

3950 An instance retrieval request using these two navigation paths with the \$refer query parameter will
 3951 again return an instance representation with two unmerged navigation properties; it is not shown for
 3952 brevity.

3953 An instance retrieval request using one of these navigation paths with the \$expand query parameter will
 3954 also return an instance representation with two unmerged navigation properties:

```
3955 GET /c1a?$expand=ACME_Assoc12.End2.ACME_Assoc23.End3&$refer=ACME_Assoc12.End2.ACME_Assoc24.End4
```

3957

```
3958 Instance c1a:
3959   Prop1: "..."/>

```

3968

```

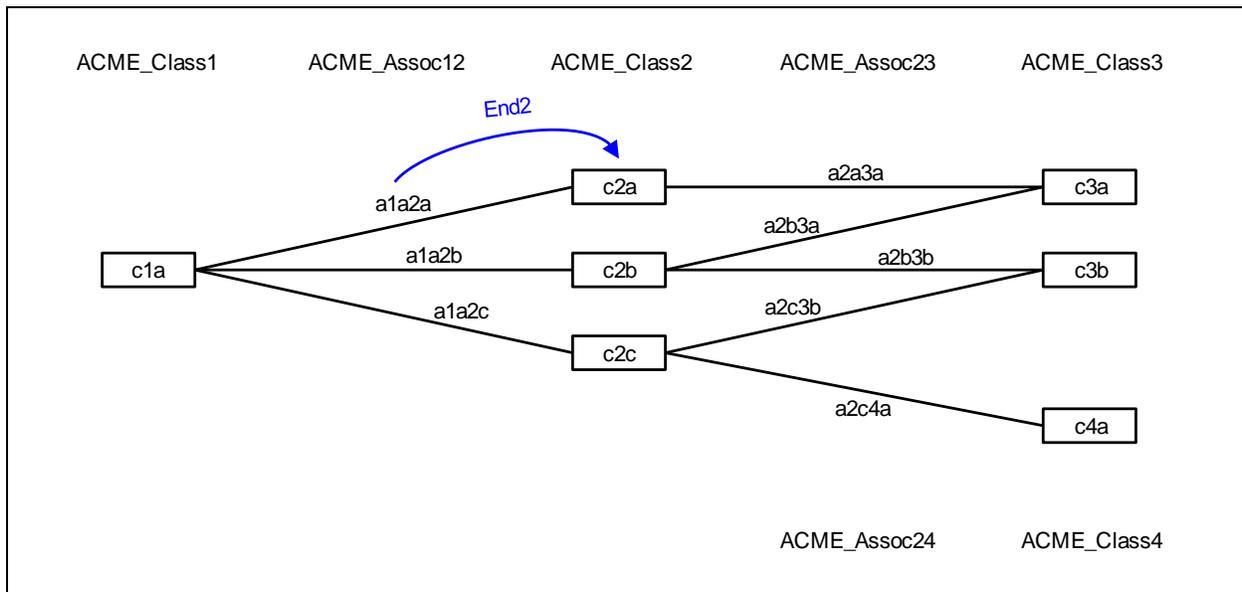
3969 ACME_Assoc12.End2.ACME_Assoc24.End4: ReferenceCollection:
3970     ref c4a
    
```

3971 The reason for not merging is that the second property would need to have an anchor point for merging
 3972 (for example, ACME_Class2 instances), and such an anchor point is not provided by the first property,
 3973 because it only represents its end of the navigation path (instances referenced by End3).

3974 This does not change even when both navigation paths are expanded, because either result is just
 3975 representing the end of the navigation without providing an anchor point for the other.

3976 **D.1.9 Expansion of association reference**

3977 In this example, the client retrieves an association instance and specifies a navigation path that expands
 3978 one of the existing references in the association. Figure D-9 shows the instance diagram and the blue
 3979 navigation path "End2", starting at instance a1a2a.



3980
3981

3982 **Figure D-9 – Example instance diagram for expansion of association reference**

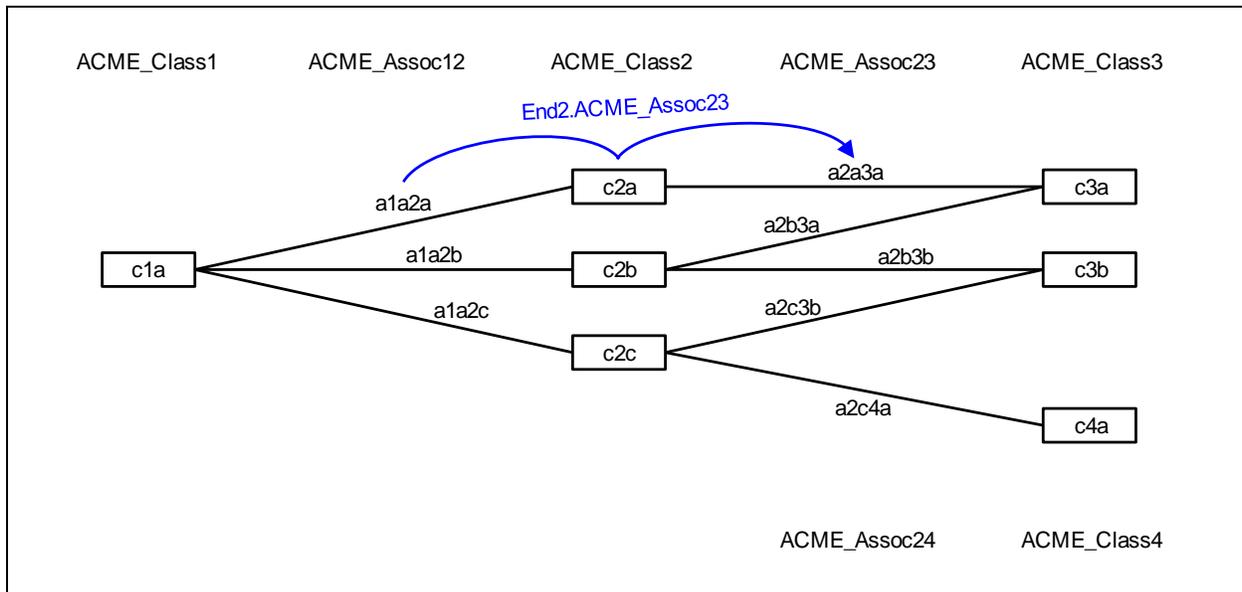
3983 An instance retrieval request using this navigation path with the \$expand query parameter will return the
 3984 following instance representation:

```

3985 GET /a1a2a?$expand=End2
3986
3987 Instance a1a2a:
3988     Prop12: "...
3989     End1: ref c1a
3990     End2: Instance c2a:
3991         Prop2: "...
    
```

3992 **D.1.10 Navigation from association to referencing association**

3993 In this example, the client retrieves an association instance and specifies a navigation path that identifies
 3994 the association instances that reference the same instances that are also referenced by the association
 3995 instance being retrieved. Figure D-10 shows the instance diagram and the blue navigation path
 3996 "End2.ACME_Assoc23", starting at instance a1a2a.



3997
3998

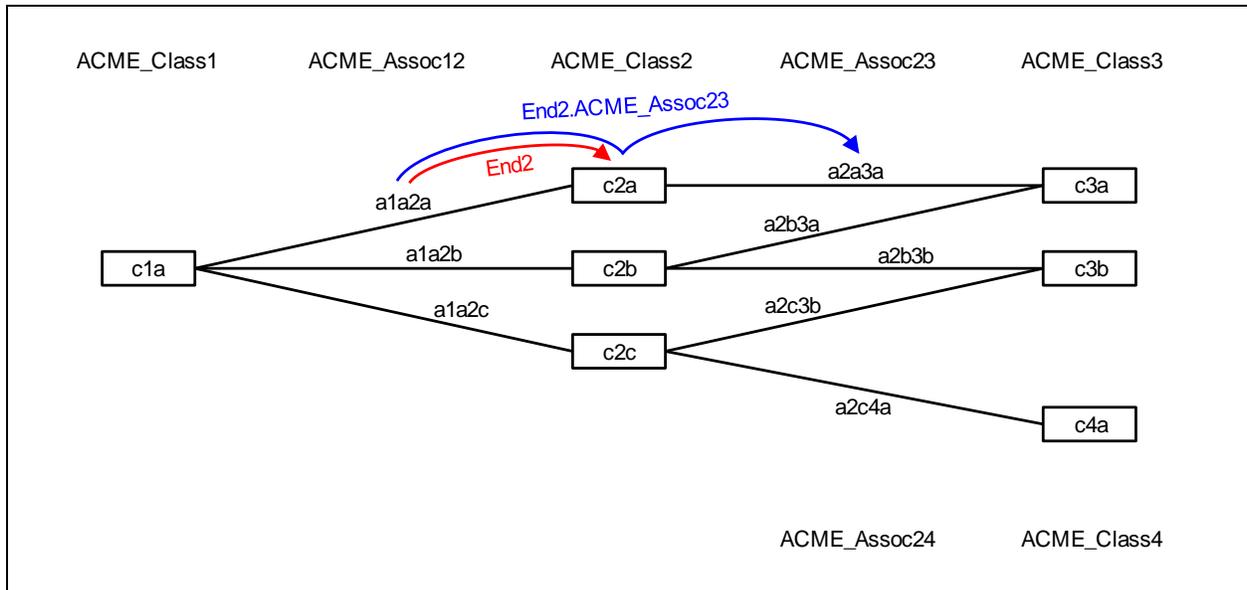
3999 **Figure D-10 – Example instance diagram for navigation starting from association**

4000 An instance retrieval request using this navigation path with the \$expand query parameter will return the
 4001 following instance representation:

```
4002 GET /a1a2a?$expand=End2.ACME_Assoc12
4003
4004 Instance a1a2a:
4005   Prop12: "...
4006   End1: ref c1a
4007   End2: ref c2a
4008   End2.ACME_Assoc12: InstanceCollection:
4009     Instance a2a3a:
4010       Prop23: "...
4011       End2: ref c2a
4012       End3: ref c3a
```

4013 **D.1.11 Expansion of association reference and navigation to referencing association**
 4014 **(merge)**

4015 In this example, the client retrieves an association instance and specifies both navigation properties from
 4016 the previous two examples. Figure D-11 shows the instance diagram, the red navigation path "End2", and
 4017 the blue navigation path "End2.ACME_Assoc23", both starting at instance a1a2a.



4018
 4019

4020 **Figure D-11 – Example instance diagram for expansion of association reference and navigation to**
 4021 **referencing association (merge)**

4022 An instance retrieval request using these navigation paths with the \$expand query parameter will return
 4023 the following instance representation:

```

4024 GET /a1a2a?$expand=End2,End2.ACME_Assoc12
4025
4026 Instance a1a2a:
4027   Prop12: "..."/>

```

4036 The two navigation paths get merged because one is a subset of the other. The inner navigation property
 4037 (specified using the navigation path "End2.ACME_Assoc12") gets merged into the existing reference
 4038 "End2" and its name gets shortened to "ACME_Assoc12" because that would be the valid navigation path
 4039 in the context of instance c2a.

4040 **EXPERIMENTAL**

4041 D.2 Paged retrieval

4042 This annex provides an example for paged retrieval, as described in 7.3.8. The example is based on the
4043 classes defined in D.1 and assumes that the client has specified a maximum size for pageable collections
4044 of 2 by using the `$max` parameter (see 6.5.5), in order to demonstrate paging with a small number of
4045 entities.

4046 Because the information that controls paging is represented in the payload, the requests and responses
4047 are shown in detail instead of using the abbreviated notation used in D.1.

4048 D.2.1 Navigation to associated instances (EXPERIMENTAL)

4049 EXPERIMENTAL

4050 The following exchange shows the example from D.1.3 that includes a navigation property with
4051 references to associated instances.

4052 Request:

```
4053 GET /cimrs/root%2Fcimv2/ACME_Class1/c1a?$refer=ACME_Assoc12.End2&$max=2 HTTP/1.1
4054 Host: server.acme.com:5988
4055 Accept: application/json;version=1.0
4056 X-CIMRS-Version: 1.0.0
```

4057 Response:

```
4058 HTTP/1.1 200 OK
4059 Date: Fri, 11 Nov 2011 10:11:00 GMT
4060 Content-Length: XXX
4061 Content-Type: application/json;version=1.0.1
4062 X-CIMRS-Version: 1.0.1
4063
4064 {
4065   "kind": "instance",
4066   "self": "/cimrs/root%2Fcimv2/ACME_Class1/c1a",
4067   "class": "ACME_Class1",
4068   "properties": {
4069     "Prop1": "...",
4070     "ACME_Assoc12.End2": {
4071       "kind": "referencecollection",
4072       "self": "/cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/1",
4073       "next": "/cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/2",
4074       "class": "ACME_Class2",
4075       "references": [
4076         "/cimrs/root%2Fcimv2/ACME_Class2/c2a",
4077         "/cimrs/root%2Fcimv2/ACME_Class2/c2b"
4078       ]
4079     }
4080   },
4081   "methods": { ... }
4082 }
```

4083 The presence of the "next" attribute in the reference collection indicates that there are more pages to
4084 retrieve, so the client issues a request to retrieve the next page of that collection:

4085 Request:

```
4086 GET /cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/2?$max=2  
4087 HTTP/1.1  
4088 Host: server.acme.com:5988  
4089 Accept: application/json;version=1.0  
4090 X-CIMRS-Version: 1.0.0
```

4091 Response:

```
4092 HTTP/1.1 200 OK  
4093 Date: Fri, 11 Nov 2011 10:11:00 GMT  
4094 Content-Length: XXX  
4095 Content-Type: application/json;version=1.0.1  
4096 X-CIMRS-Version: 1.0.1  
4097  
4098 {  
4099   "kind": "referencecollection",  
4100   "self": "/cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/2",  
4101   "class": "ACME_Class2",  
4102   "references": [  
4103     "/cimrs/root%2Fcimv2/ACME_Class2/c2c"  
4104   ]  
4105 }
```

4106 This time, the reference collection does not contain a next attribute, indicating that the collection is now
4107 complete.

4108 The variant using the \$expand parameter is omitted; paged retrieval works the same for that variant
4109 except that the response now contains an instance collection instead of the reference collection. See
4110 7.8.2 for an example of an instance collection retrieval.

4111 **EXPERIMENTAL**

4112
4113
4114
4115**ANNEX E**
(informative)**Change log**

Version	Date	Description
1.0.0	2013-01-24	
1.0.1	2014-02-11	Released as DMTF Standard, with the following changes: <ul style="list-style-type: none">• Changed the concept of navigation paths and the \$expand and \$refer query parameters back to experimental• Added statement that examples use the payload representation from DSP0211• Removed incorrect attribution of instance and reference collections to listeners in Table 2• Changed to use new generic operation names• Fixed incorrectly named query parameters• Fixed editorial issues with table and figure naming

Bibliography

4116

4117 This annex contains a list of non-normative references for this document.

4118 DMTF DSP0200, *CIM Operations over HTTP 1.3*,

4119 http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf

4120 DMTF DSP1001, *Management Profile Specification Usage Guide 1.1*,

4121 http://www.dmtf.org/standards/published_documents/DSP1001_1.1.pdf

4122 DMTF DSP1033, *Profile Registration Profile 1.0*,

4123 http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf

4124 DMTF DSP1054, *Indications Profile 1.2*,

4125 http://www.dmtf.org/sites/default/files/standards/documents/DSP1054_1.2.pdf

4126 DMTF DSP2032, *CIM-RS White Paper 1.0*,

4127 http://www.dmtf.org/standards/published_documents/DSP2032_1.0.pdf

4128 ECMA-262, *ECMAScript Language Specification, 5th Edition*, December 2009,

4129 <http://www.ecma-international.org/publications/standards/Ecma-262.htm>

4130 IETF RFC2608, *Service Location Protocol, Version 2*, June 1999,

4131 <http://tools.ietf.org/html/rfc2608>

4132 IETF RFC4648, *The Base16, Base32, and Base64 Data Encodings*, October 2006,

4133 <http://tools.ietf.org/html/rfc4648>

4134 IETF RFC5005, *Feed Paging and Archiving*, September 2007,

4135 <http://tools.ietf.org/html/rfc5005>

4136 IETF Draft RFC *Additional HTTP Status Codes*, Draft 04, February 2012,

4137 <http://tools.ietf.org/html/draft-nottingham-http-new-status-04>

4138 IANA Permanent Message Header Field Names,

4139 <http://www.iana.org/assignments/message-headers/perm-headers.html>

4140 IANA MIME Media Types,

4141 <http://www.iana.org/assignments/media-types/>

4142 ITU-T X.509, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*,

4143 <http://www.itu.int/rec/T-REC-X.509/en>

4145 R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, PhD thesis, University of California, Irvine, 2000,

4147 <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

4148 R. Fielding, *REST APIs must be hypertext driven*, October 2008,

4149 <http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven>

4150 J. Holzer, *RESTful Web Services and JSON for WBEM Operations*, Master thesis, University of Applied Sciences, Konstanz, Germany, June 2009,

4152 <http://mond.htwg-konstanz.de/Abschlussarbeiten/Details.aspx?id=1120>

4153 A. Manes, *Rest principle: Separation of representation and resource*, March 2009,

4154 <http://apsblog.burtongroup.com/2009/03/rest-principle-separation-of-representation-and-resource.html>

4155 L. Richardson and S. Ruby, *RESTful Web Services*, May 2007, O'Reilly, ISBN 978-0-596-52926-0,

4156 <http://www.oreilly.de/catalog/9780596529260/>