1

5   # CIM-RS Protocol

11

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

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28 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

CONTENTS

195   **Figures**

199   **Tables**

250

251 # Foreword

268   # Introduction

269   The information in this document should be sufficient to unambiguously identify the protocol interactions
270   that shall be supported when implementing the CIM-RS protocol. The CIM-RS protocol follows the
271   principles of the REST architectural style for accessing modeled resources whose model conforms to the
272   CIM metamodel defined in DSP0004.

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

275   ## Document conventions

276   ### Typographical conventions

277   The following typographical conventions are used in this document:

278   - Document titles are marked in *italics*.

279   - ABNF rules and JSON text are in `monospaced font`.

280   ### ABNF usage conventions

281   Format definitions in this document are specified using ABNF (see RFC5234), with the following
282   deviations:

283   - Literal strings are to be interpreted as case-sensitive UCS characters, as opposed to the
284   definition in RFC5234 that interprets literal strings as case-insensitive US-ASCII characters.

285 <h1 style="text-align:center">CIM-RS Protocol</h1>

286 # 1   Scope

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

289 REST architectural style was first described by Roy Fielding in chapter 5 of *Architectural Styles and the*
290 *Design of Network-based Software Architectures* and in *REST APIs must be hypertext driven*. This style
291 generally results in simple interfaces that are easy to use and that do not impose a heavy burden on
292 client side resources.

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

295 The DMTF base requirements for interoperable communication between WBEM clients and servers are
296 defined collectively by DSP0004 and DSP0223. These specifications form the basis for profiles (see
297 DSP1001) that define interfaces for specific management purposes.

298 The semantics of CIM-RS protocol operations are first described in a standalone manner and then are
299 mapped to the generic operations defined in DSP0223.

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

304 The CIM-RS protocol can be used with HTTP and HTTPS. Unless otherwise noted, the term HTTP in this
305 document refers to both HTTP and HTTPS.

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

310 Background information for CIM-RS is described in a white paper, DSP2032.

311 # 2   Normative references

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

316 DMTF DSP0004, *CIM Infrastructure Specification 2.8*,
317 http://www.dmtf.org/standards/published_documents/DSP0004_2.8.pdf

318 DMTF DSP0198, *WBEM Glossary 1.0*,
319 http://www.dmtf.org/standards/published_documents/DSP0198_1.0.pdf

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

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

324 DMTF DSP0207, *WBEM URI Mapping 1.0*,
325 http://www.dmtf.org/standards/published_documents/DSP0207_1.0.pdf

326 DMTF DSP0211, *CIM-RS Payload Representation in JSON 2.0*,
327 http://www.dmtf.org/standards/published_documents/DSP0211_2.0.pdf

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

330 DMTF DSP0223, *Generic Operations 2.0*,
331 http://www.dmtf.org/standards/published_documents/DSP0223_2.0.pdf

332 IETF RFC2246, *The TLS Protocol Version 1.0*, January 1999,
333 http://tools.ietf.org/html/rfc2246

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

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

338 IETF RFC2818, *HTTP Over TLS*, May 2000,
339 http://tools.ietf.org/html/rfc2818

340 IETF RFC3986, *Uniform Resource Identifier (URI): Generic Syntax*, January 2005,
341 http://tools.ietf.org/html/rfc3986

342 IETF RFC4346, *The Transport Layer Security (TLS) Protocol, Version 1.1*, April 2006,
343 http://tools.ietf.org/html/rfc4346

344 IETF RFC5234, *Augmented BNF for Syntax Specifications: ABNF*, January 2008,
345 http://tools.ietf.org/html/rfc5234

346 IETF RFC5246, *The Transport Layer Security (TLS) Protocol, Version 1.2*, August 2008,
347 http://tools.ietf.org/html/rfc5246

348 IETF RFC6585 *Additional HTTP Status Codes*, April 2012,
349 http://tools.ietf.org/html/rfc6585

350 IETF RFC6838, *Media Type Specifications and Registration Procedures*, January 2013,
351 http://tools.ietf.org/html/rfc6838

352 IETF RFC6839, *Additional Media Type Structured Syntax Suffixes*, January 2013,
353 http://tools.ietf.org/html/rfc6839

354 ISO/IEC 10646:2003, *Information technology -- Universal Multiple-Octet Coded Character Set (UCS)*,
355 http://standards.iso.org/ittf/PubliclyAvailableStandards/c039921_ISO_IEC_10646_2003(E).zip

356 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards (2004, 5th
357 edition)*,
358 http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse

359 NIST Special Publication 800-57, Elaine Barker et al, *Recommendation for Key Management – Part 1:
360 General (Revised)*, March 2007,
361 http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf

362 NIST Special Publication 800-131A*,* Elaine Barker and Allen Roginsky, *Transitions: Recommendation for*
363 *Transitioning the Use of Cryptographic Algorithms and Key Lengths*, January 2011,
364 http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

365 The Unicode Consortium, The Unicode Standard, Version 5.2.0, Annex #15: Unicode Normalization
366 Forms,
367 http://www.unicode.org/reports/tr15/

3   Terms and definitions

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

371 The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"),
372 "may", "need not" ("not required"), "can", and "cannot" in this document are to be interpreted as described
373 in ISO/IEC Directives, Part 2, Annex H. The terms in parenthesis are alternatives for the preceding term,
374 for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that
375 ISO/IEC Directives, Part 2, Annex H specifies additional alternatives. Occurrences of such additional
376 alternatives shall be interpreted in their normal English meaning.

377 The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as
378 described in ISO/IEC Directives, Part 2, clause 5.

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

382 The terms defined in DSP0198 apply to this document. Specifically, this document uses the terms
383 "model", "namespace", "qualifier", "qualifier type", "class", "creation class", "ordinary class", "association",
384 "indication", "instance", "property", "ordinary property", "reference", "method", "parameter", "WBEM client"
385 ("client"), "WBEM server" ("server"), and "WBEM listener" ("listener") defined in DSP0198.

386 The following additional terms are used in this document.

387 **3.1**
388 **CIM-RS payload data type**
389 a data type for CIM-RS payload elements, or components thereof. Also called "payload data type" in this
390 document. Payload data types are abstractly defined in this document, and concretely in CIM-RS payload
391 representation specifications, and are thus part of the interface between these documents. For the list of
392 payload data types defined for the CIM-RS protocol, see Table 5.

393 **3.2**
394 **CIM-RS operation**
395 an interaction in the CIM-RS protocol where a WBEM client invokes an action in a WBEM server, or a
396 WBEM server invokes an action in a WBEM listener. For a full definition, see 5.1.

397 **3.3**
398 **CIM-RS payload element**
399 a particular kind of content of the entity body of the HTTP messages used by the CIM-RS protocol. Also
400 called "payload element" in this document. Payload elements are abstractly defined in this document, and
401 concretely in CIM-RS payload representation specifications, and are thus part of the interface between
402 these documents. For the list of payload elements defined for the CIM-RS protocol, see Table 4.

403   **3.4**
404   **CIM-RS payload representation**

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

408   **3.5**
409   **CIM-RS payload representation specification**

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

411   **3.6**
412   **CIM-RS protocol**

413   the RESTful protocol defined in this document.

414   **3.7**
415   **CIM-RS resource**

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

419   **3.8**
420   **CIM-RS resource identifier**

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

423   **3.9**
424   **HTTP basic authentication**

425   a simple authentication scheme for use by HTTP and HTTPS that is based on providing credentials in
426   HTTP header fields. It is defined in RFC2617.

427   **3.10**
428   **HTTP content negotiation**

429   a method for selecting a representation of content in an HTTP response message when there are multiple
430   representations available. It is defined in section 12 of RFC2616. Its use in the CIM-RS protocol is
431   described in 7.3.1.

432   **3.11**
433   **HTTP digest authentication**

434   an authentication scheme for use by HTTP and HTTPS that is based on verifying shared secrets that are
435   not exchanged. It is defined in RFC2617.

436   **3.12**
437   **HTTP entity body**

438   the payload within an HTTP message, as defined in section 7.2 of RFC2616.

439   **3.13**
440   **HTTP entity-header field**

441   a header field that may be used in HTTP requests and HTTP response messages, specifying information
442   that applies to the data in the entity body. Also called "HTTP entity-header".

443 **3.14**
444 **HTTP extension-header field**

445 an entity-header field used for custom extensions to the standard set of header fields defined in
446 RFC2616. Also called "HTTP extension-header".

447 **3.15**
448 **HTTP general-header field**

449 a header field that may be used in HTTP requests and HTTP response messages, specifying information
450 that applies to the HTTP message. Also called "HTTP general-header".

451 **3.16**
452 **HTTP header field**

453 a named value used in the header of HTTP messages, as defined in section 4.2 of RFC2616. Also called
454 "HTTP header". The specific types of header fields are general-header field, request-header field,
455 response-header field, entity-header field, and extension-header field.

456 **3.17**
457 **HTTP message**

458 an interaction between an HTTP client and an HTTP server (in any direction), as defined in section 4 of
459 RFC2616.

460 **3.18**
461 **HTTP method**

462 the type of interaction stated in HTTP requests, as defined in section 5.1.1 of RFC2616.

463 **3.19**
464 **HTTP request message**

465 an HTTP message sent from an HTTP client to an HTTP server as defined in section 5 of RFC2616. Also
466 called "HTTP request".

467 **3.20**
468 **HTTP request-header field**

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

471 **3.21**
472 **HTTP response message**

473 an HTTP message sent from an HTTP server to an HTTP client, as defined in section 6 of RFC2616. Also
474 called "HTTP response".

475 **3.22**
476 **HTTP response-header field**

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

479 **3.23**
480 **Internet media type**

481 a string identification for representation formats in Internet protocols. Originally defined for email
482 attachments and termed "MIME type". Because the CIM-RS protocol is based on HTTP, it uses the
483 definition of media types from section 3.7 of RFC2616.

3.24

Interop namespace

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

3.25

Normalization Form C

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

3.26

reference-qualified property

a string-typed CIM property qualified with the *Reference* qualifier (see DSP0004 for a definition of the *Reference* qualifier, and 5.3.3 for details).

3.27

reference-typed parameter

a CIM method parameter declared with a CIM data type that is a reference to a specific class.

3.28

reference-typed property

a CIM property declared with a CIM data type that is a reference to a specific class. See 5.3.3 for details. DSP0004 defines the term "reference" for such properties; this document uses the more specific term "reference-typed property", instead.

3.29

reference property

a general term for reference-typed properties and reference-qualified properties. See 5.3.3 for details.

3.30

reserved character

a character from the set of *reserved* characters defined for URIs in RFC3986. See 6.3 for details.

3.31

resource representation

a representation of a resource or some aspect thereof, in some format. A particular resource may have any number of representations. The format of a resource representation is identified by a media type. In the CIM-RS protocol, the more general term "payload representation" is used, because not all payload elements are resource representations.

3.32

REST architectural style

the architectural style described in *Architectural Styles and the Design of Network-based Software Architectures*, chapter 5, and in *REST APIs must be hypertext driven*.

3.33

UCS character

a character from the Universal Character Set defined in ISO/IEC 10646:2003. See also DSP0004 for the usage of UCS characters in CIM strings. An alternative term is "Unicode character".

526 **3.34**
527 **unreserved character**
528 a character from the set of *unreserved* characters defined for URIs in RFC3986. See 6.3 for details.

4 Symbols and abbreviated terms

530 The abbreviations defined in DSP0198 apply to this document. Specifically, this document uses the
531 abbreviations "ABNF", "CIM", "FQL", "HTTP", "IANA", "REST", "SLP", "UCS", "URI", "WBEM", and "XML"
532 defined in DSP0198.

533 The following additional abbreviations are used in this document.

534 **4.1**
535 **CIM-RS**
536 **CIM RESTful Services**
537 the name of the protocol defined in this document and related documents.

538 **4.2**
539 **HTTPS**
540 Hyper Text Transfer Protocol Secure, as defined in RFC2818.

541 **4.3**
542 **JSON**
543 JavaScript Object Notation, as defined in RFC7159.

544 **4.4**
545 **UTF-8**
546 UCS Transformation Format 8, as defined in ISO/IEC 10646:2003.

5 Concepts

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

5.1 CIM-RS protocol participants

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

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

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

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

560
561
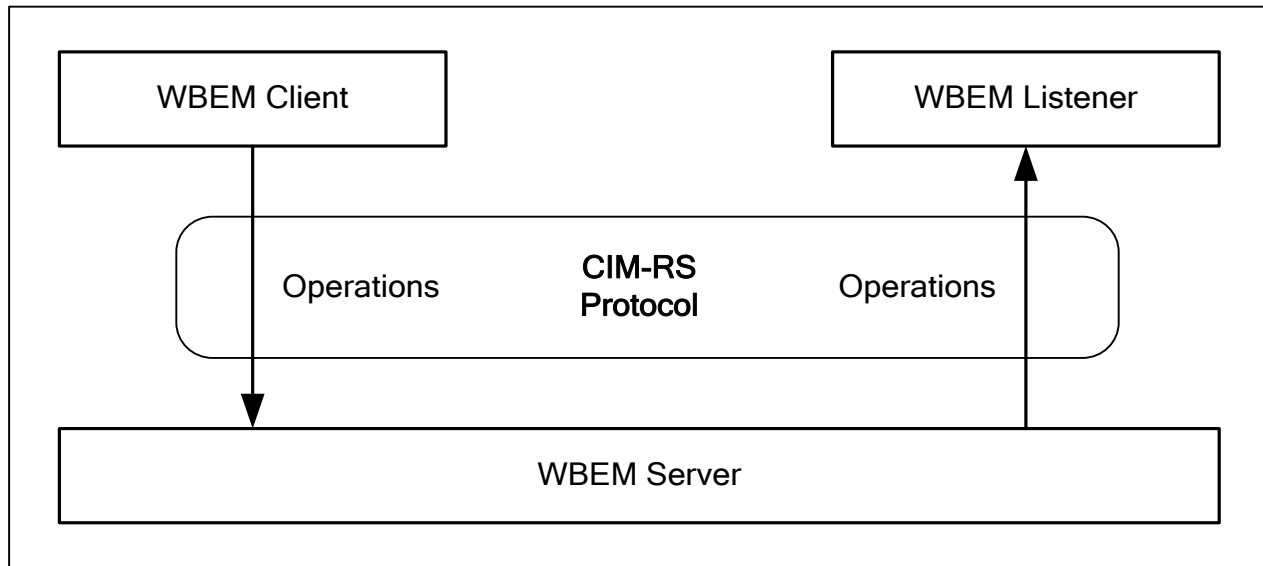
Figure 1 – Participants in the CIM-RS protocol

5.2   Model independence of CIM-RS

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

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

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

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

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

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

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

599
600
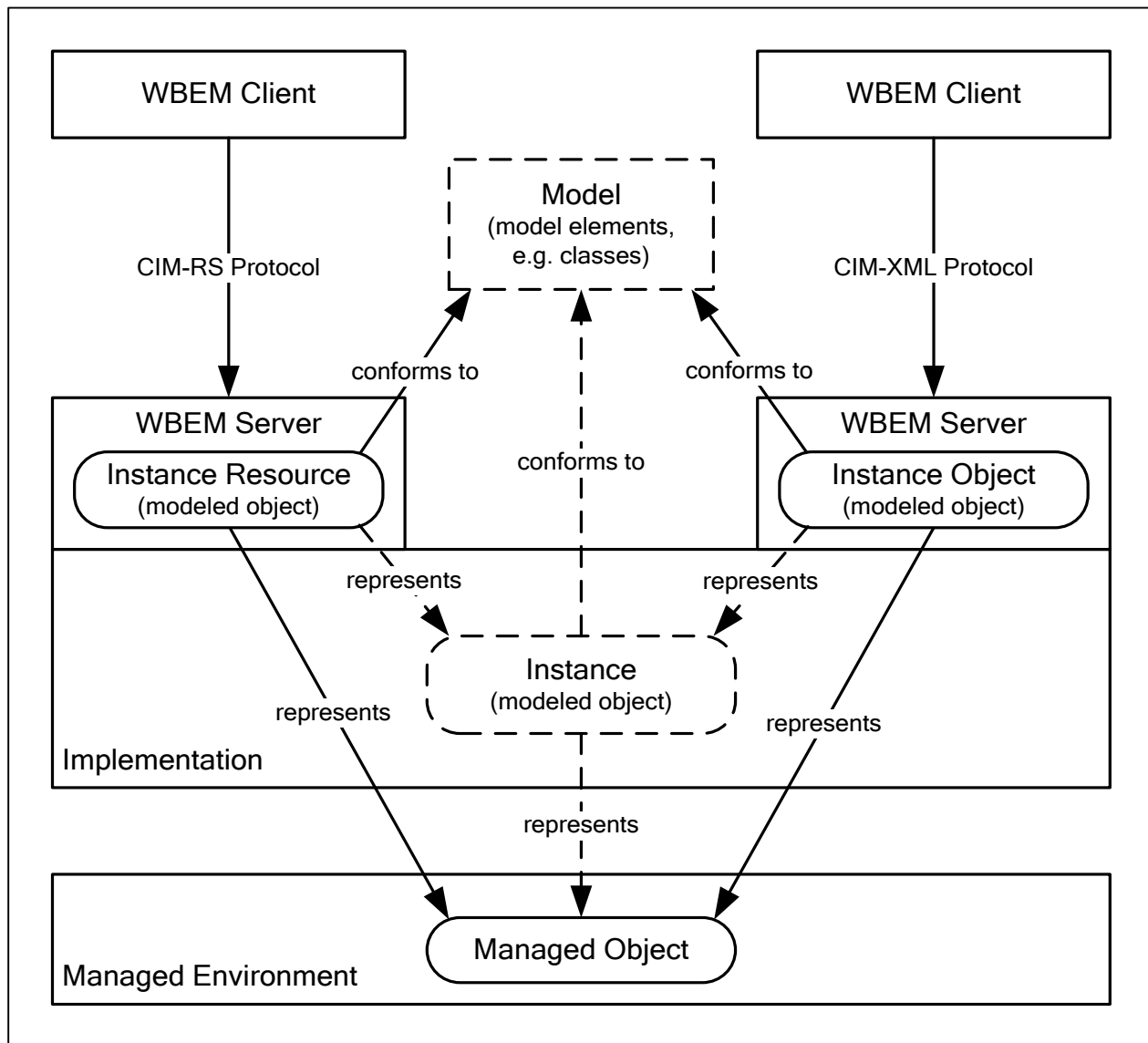
601   **Figure 2 – Single model and multiple protocols**

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

5.3   Mapping model elements to CIM-RS resources (informative)

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

5.3.1   Classes

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

Classes are represented as CIM-RS resources; more specifically as *class resources* (see 7.10).

5.3.2   Instances

Addressable instances of classes are represented as CIM-RS resources; more specifically as *instance resources* (see 7.5).

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

Behaviors of instances are the class-defined (extrinsic) methods and certain built-in (intrinsic) operations; they are represented as HTTP methods either directly on the instance resource, or on the class resource of the creation class of the instance.

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

5.3.3   Properties

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

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

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

- reference-typed properties – These are reference properties in association classes that are declared with a CIM data type that is a reference to a specific class; they are the ends of associations. Reference-typed properties are always scalars; there are no arrays of reference-typed properties. The value of a reference-typed property references a single instance.

- reference-qualified properties – These are string-typed properties that are qualified with the *Reference* qualifier. These properties can be used in ordinary classes; they are like simple pointers to instances and do not constitute association ends or imply any associations. Reference-qualified properties may be scalars or arrays. The value of a reference-qualified scalar property and the value of an array entry of a reference-qualified array property reference a single instance.

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

5.3.4   Methods and operations

Class-defined (extrinsic) methods can be defined as being static or non-static. Non-static methods are invoked via HTTP POST on an instance resource (see 7.5.8). Static methods are invoked via HTTP POST on a class resource (see 7.10.6) or an instance resource (see 7.5.8).

645 CIM-RS supports a set of built-in operations that are not class-defined. These operations are the typical
646 CRUD (Create, Read, Update, and Delete) operations of REST environments; they are invoked by means
647 of HTTP methods: POST, GET, PUT, and DELETE directly on the instance resource for reading, updating
648 and deleting, respectively (see 7.10.6).

649 5.4 Two-staged mapping approach

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

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

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

657 Figure 3 shows a pictorial representation of this two-staged mapping approach:
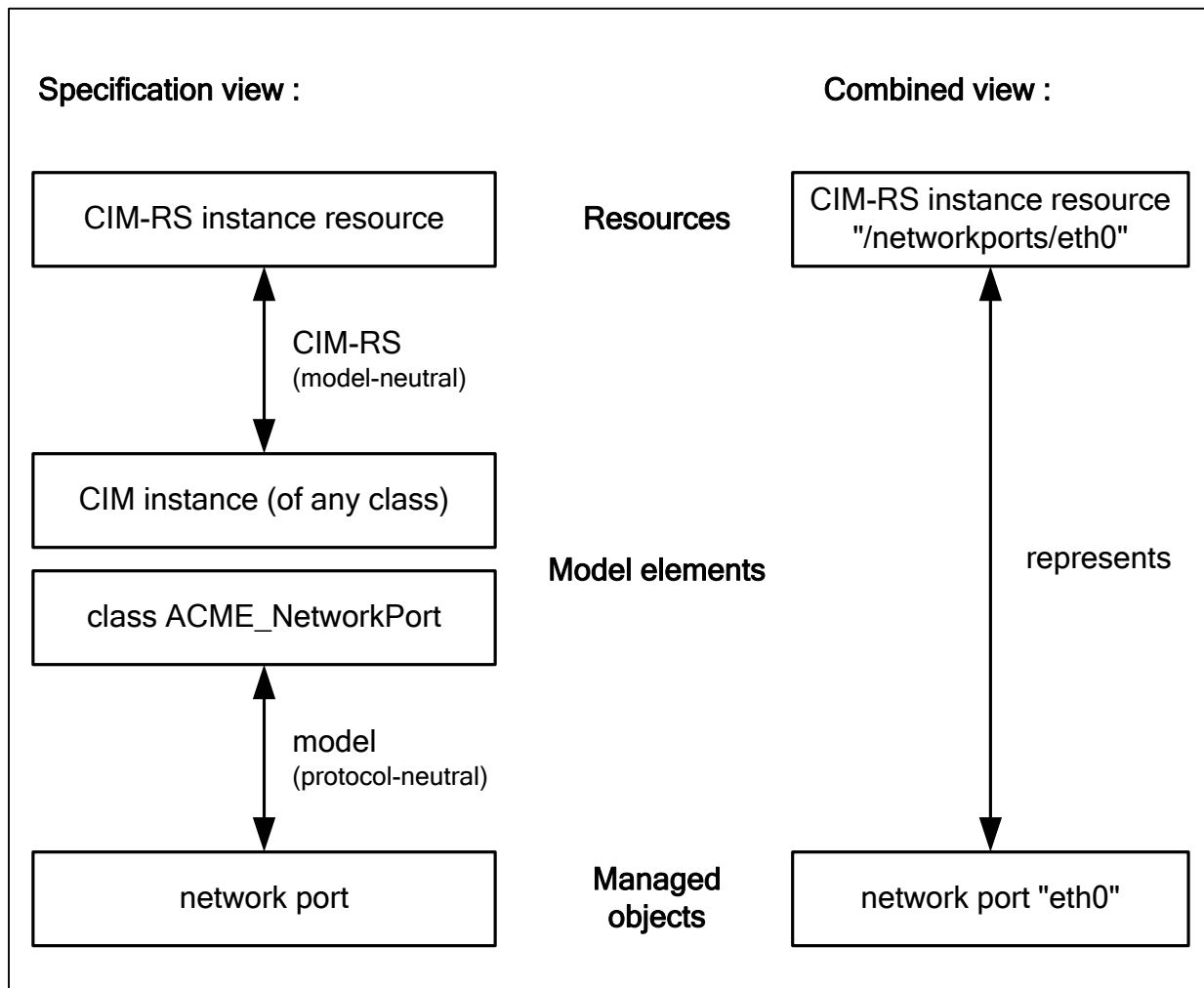
658

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

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

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

5.5   REST architectural style supported by CIM-RS

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

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

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

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

676   NOTE: Pulled enumeration operations as defined in DSP0223 maintain the enumeration state either on
677   the server side or on the client side. In both approaches, the client needs to hand back and forth an
678   opaque data item called enumeration context, which is the actual enumeration state in case of a client-
679   maintained enumeration state, or a handle to the enumeration state in case of a server-maintained
680   enumeration state. CIM-RS supports both of these approaches. It is possible for a server to remain
681   stateless as far as the enumeration state goes, by implementing the client-based approach. The approach
682   implemented by a server is not visible to a client, because the enumeration context handed back and forth
683   is opaque to the client in both approaches.

684 • **Cache:** The HTTP methods used by CIM-RS are used as defined in RFC2616. As a result, they
685   are cacheable as defined in RFC2616. This constraint is fully satisfied.

686   NOTE   RFC2616 defines only the result of HTTP GET methods to be cacheable.

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

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

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

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

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

708 The REST architectural style recommends that all addressing information for a resource is in the resource
709 identifier (and not, for example, in the HTTP header). In addition, it recommends that resource identifiers
710 are opaque to clients and clients should not be required to understand the structure of resource identifiers
711 or be required to assemble any resource identifiers. CIM-RS follows these recommendations. Even
712 though resource identifiers in CIM-RS are well-defined and are not opaque to clients, clients are not
713 required to understand the structure of resource identifiers and are not required to assemble any resource
714 identifiers.

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

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

724 Clients can completely discover any resources in a WBEM server, and even the server itself. See 7.18 for
725 details on typical discovery related interactions.

6   Resource identifiers

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

731 Usages of the resource identifier URI in the HTTP header are defined in RFC2616 and RFC2818. In the
732 protocol payload, resource identifiers are values of type URI (see Table 5), using the format defined in
733 6.1.

6.1   CIM-RS resource identifier format

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

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

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

738 Where:

739 • `scheme` is defined in RFC3986 and shall in addition conform to the definitions in 6.4

740 • `authority` is defined in RFC3986 and shall in addition conform to the definitions in 6.5

741 • `path-absolute` is defined in RFC3986

742 • `query` is defined in RFC3986 and shall in addition conform to the definitions in 6.6

743 This format conforms to but restricts ABNF rule `URI-reference` defined in RFC3986.

744 RFC3986 defines the concept of a base URI that can be used to have shorter URIs relative to the base
745 URI. The base URI for CIM-RS resource identifiers referencing resources in a server or listener shall be
746 the absolute URI of the server or listener, respectively. In other words, CIM-RS resource identifiers that
747 are relative to such a base URI conform to the ABNF rule `cimrs-uri-based`:

748   `cimrs-uri-based = path-absolute [ "?" query ]`

749   The scheme component in CIM-RS resource identifiers may be present, but is not needed in CIM-RS
750   resource identifiers because they are intended to be independent of the access protocol (HTTP or
751   HTTPS). Specifying any supported scheme or omitting it does not affect the identification of the resource.

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

756   The use of fragments is not permitted in CIM-RS resource identifiers because resource identifiers serve
757   the purpose of identifying resources, and fragments are not part of the resource identification (see
758   RFC3986).

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

761   CIM-RS resource identifiers shall conform to the rules on URLs/URIs defined in RFC2616 (for HTTP) and
762   RFC2818 (for HTTPS).

763   6.2   Non-opaqueness

764   CIM-RS resource identifiers are generally non-opaque, in the sense that their format is well-defined. For
765   details, see clause 7. As a result, resource identifiers may be parsed, constructed or modified, as needed.

766   Specifically, the following changes to resource identifiers are typical:

767   • Parsing, adding, removing or modifying any query parameters in a resource identifier

768   • Normalizing a resource identifier, as described in RFC3986 (for example, removing ".." and "."
769   segments)

770   Note that some resource identifiers or components thereof are specific to the server implementation and
771   thus cannot be constructed, parsed, or modified by clients:

772   • Resource identifiers of instance collection page resources are server-implementation-specific.

773   • Key bindings in the resource identifiers of CIM instances may be specific to the class-specific
774   implementation, and may not be predictable for clients.

775   6.3   Percent-encoding

776   This subclause defines how the percent-encoding rules defined in RFC3986 are applied to resource
777   identifiers.

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

779   •   *Unreserved* characters should not be percent-encoded. If they are percent-encoded, consumers
780   of the resource identifier shall tolerate that. Unreserved characters are defined in ABNF rule
781   `unreserved` in RFC3986 as follows:

```
782   unreserved  = ALPHA / DIGIT / "-" / "." / "_" / "~"
783
784   ALPHA = %x41-5A / %x61-7A
785
786   DIGIT = %x30-39
```

787   •   The percent-encoding of *reserved* characters depends on whether the character in question is
788   considered a delimiter or data.

789   Reserved characters are defined in ABNF rule `reserved` in RFC3986 as follows:

```
790   reserved  = gen-delims / sub-delims
791
792   gen-delims    = ":" / "/" / "?" / "#" / "[" / "]" / "@"
793
794   sub-delims    = "!" / "$" / "&" / "'" / "(" / ")"
795                   / "*" / "+" / "," / ";" / "="
```

796   Reserved characters that are considered delimiters shall not be percent-encoded.

797   Reserved characters that are considered data shall be percent-encoded.

798   The definitions of query parameters in 6.6 and resource identifiers in clause 7 state which of the
799   reserved characters are considered delimiters or data, for purposes of percent-encoding.

800   •   Any other characters (that is, outside of the ABNF rules `reserved` and `unreserved` defined in
801   RFC3986) shall be percent-encoded.

802   Consumers of resource identifiers shall support any percent-encoding within the resource identifier that is
803   permissible according to the rules in this subclause.

804   RFC3986 defines percent-encoding on the basis of data octets, but it does not define how characters are
805   encoded as data octets. Because element names, namespace names, and key values may contain UCS
806   characters outside of the US-ASCII character set, this document defines the percent-encoding to be used
807   in resource identifiers as follows.

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

814   If values of CIM data types need to be represented in resource identifiers, the data type-specific string
815   representations defined in DSP0004 shall be used.

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

817   •   The namespace name "root/cimv2" becomes "root%2Fcimv2" in a resource identifier, because
818   the slash character (/) is a reserved character in resource identifiers and the subclauses on
819   resource identifiers state that an occurrence of a slash in a namespace name is considered
820   data.

821   •   The class name "ACME_LogicalDevice" remains unchanged in a resource identifier, because it
822   contains only unreserved characters.

823   •   The (German) key property value "ÄnderungsRate" becomes "%C3%84%0AnderungsRate" in a
824   resource identifier, because C3 84 0A are the data octets of the UTF-8 encoding of the UCS
825   character U+00C4, which represents "Ä" (A umlaut) in normalized form. Note that usage of the
826   UCS character sequence U+0061 U+0308 which also represents "Ä" (using the base character
827   "A" and the combining diacritical mark ¨) is not permitted due to the requirement to use
828   Normalization Form C.

829   •   The string-typed value "a \"brown\" bag\n" (represented using backslash escape sequences as
830   defined for string literals in MOF) becomes "a%20%22brown%22%20bag%0A" in a resource
831   identifier, because the characters blank (U+0020), newline (U+000A), and double quote
832   (U+0022) are not in the ABNF rules `reserved` and `unreserved` defined in RFC3986, and
833   therefore need to be percent-encoded.

834   •   The sint8-typed value -42 becomes the string "-42" in a resource identifier, because that is the
835   string representation of an sint8-typed value defined in DSP0004, and because "-" is an
836   unreserved character that has been chosen not to be percent-encoded in order to produce a
837   minimally percent-encoded URI.

6.4   Scheme component

839   WBEM clients, servers, and listeners shall adhere to the following additional rules regarding the value of
840   ABNF rule `scheme` defined in 6.1:

841   •   The rules for the scheme component defined in RFC2616 (for HTTP) and RFC2818 (for
842   HTTPS) apply.

843   As a result, the only permitted scheme values are `"http"` and `"https"` (and their variations in lexical
844   case).

6.5   Authority component

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

848   •   The `userinfo` component within `authority` shall not be specified because of security issues
849   with specifying an unencrypted password

850   •   The `host` component within `authority` shall be the IP (V4 or V6) address of the server, or a
851   DNS-resolvable host name for that IP address (including `"localhost"`)

852   •   If the `port` component within `authority` is not specified, the port number shall default to the
853   standard port numbers for CIM-RS:

854   –   port number 5993 when using HTTP

855   –   port number 5994 when using HTTPS

856   Note that these port numbers have been requested but are not approved by IANA at the time of
857   release of this document. See the IANA Port Number Registry for approved port numbers.

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

861   6.6   Query parameters

862   This subclause defines the query component of resource identifiers, and applies in addition to the
863   definition in RFC3986, section 3.4.

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

865   ```
query = query-parameter *("&" query-parameter)
```

866   Where:

867   • `query-parameter` is a query parameter as defined in the subclauses of this subclause

868   • The reserved character "&" in the literals of this ABNF rule shall be considered a delimiter for
869       purposes of percent-encoding (see 6.3, that is, it shall not be percent-encoded).

870   Example:

871   • `$class=ACME_ComputerSystem&$subclasses=true`

872       This query component specifies the query parameters `$class` with a value of
873       `ACME_ComputerSystem` and `$subclasses` with a value of `true`

874   • `$properties=Name,Caption`

875       This query component specifies the query parameter `$property` with a value of
876       `Name,Caption`. The comma (,) in that value is not percent-encoded because the definition of
877       the `$properties` query parameter (see 6.6.10) states that it is considered a delimiter.

878   • `$filter=Name%3D%27a%26b%27`

879       This query component specifies the query parameter `$filter` with a value of `Name='a&b'`,
880       percent-encoding the reserved characters "=", ampersand (&), and single quote (') because the
881       definition of the `$filter` query parameter (see 6.6.6) states that they are considered data.

882   Query parameters of resource identifiers (that is, both name and value) are case sensitive, as defined in
883   RFC3986, section 6.2.2.1, unless defined otherwise in this subclause. The query parameters defined in
884   the subclauses of this subclause define in some cases that the values of query parameters are to be
885   treated case insensitively. In such cases, two resource identifiers that differ only in the lexical case of
886   query parameters address the same resource, even though the resource identifiers do not match
887   according to the rules defined in RFC3986. It is recommended that producers of resource identifiers
888   preserve the lexical case in such case insensitive cases, in order to optimize caching based on resource
889   identifiers. For example, if a property is named "ErrorRate", its use in the `$properties` query parameter
890   should be "`$properties=ErrorRate`", preserving its lexical case.

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

897   NOTE    RFC3986 does not detail how the `query` ABNF rule is broken into query parameters, and thus does not
898   address the topic of query parameter repetition.

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

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

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

908  ```
query-parameter = ""   ; initially empty
```

909 Table 1 lists the query parameters that are defined in CIM-RS. All those query parameters shall be
910 supported (that is, implemented) by the WBEM server. Their use in URIs is always optional in CIM-RS.
911 For details, see the subclauses on the individual operations in clause 7.

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

Query Parameter	Purpose	Description
`$associatedclass`	associated class filter	see 6.6.1
`$associatedrole`	associated role filter	see 6.6.2
`$associationclass`	association class filter	see 6.6.3
`$class`	specify class name	see 6.6.4
`$continueonerror`	continue on errors within paged retrieval	see 6.6.5
`$filter`	filter instances in result	see 6.6.6
`$filterlanguage`	specify filter language for `$filter`	see 6.6.7
`$max`	limit number of collection members in result	see 6.6.8
`$pagingtimeout`	specify inactivity timeout for paged retrieval	see 6.6.9
`$properties`	subset properties in result	see 6.6.10
`$qualifiers`	include qualifiers in returned classes	see 6.6.11
`$sourcerole`	source role filter	see 6.6.12
`$subclasses`	include subclasses in class enumeration result	see 6.6.13

913 Additional implementation-defined query parameters are not permitted in CIM-RS.

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

6.6.1 $associatedclass (associated class filter)

919 The `$associatedclass` query parameter is used to specify a filter in association traversal operations
920 that filters the result on the name of the associated class. The details of the semantics are described in
921 the association traversal operations (see 7.7.2, 7.8.2, 7.12.2, and 7.13.2).

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

923   `query-parameter =/ associatedclass-query-parm`
924
925   `associatedclass-query-parm = "$associatedclass=" class-name`

926   Where:

927   • The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
928   delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)

929   • `class-name` is the name of the associated class (including schema prefix). Note that CIM
930   class names do not contain reserved characters (see 6.3 and [DSP0004](#))

931   The `$associatedclass` query parameter shall not be repeated in a resource identifier.

932   Examples:

933   (not specified)

934   specifies no filtering on the associated class name

935   `$associatedclass=ACME_Device`

936   specifies filtering on the associated class name "ACME_Device"

937   6.6.2 $associatedrole (associated role filter)

938   The `$associatedrole` query parameter is used to specify a filter in association traversal operations
939   that filters the result on the role name for the associated class; that is, the name of the reference property
940   in the traversed association that references the associated (= far end) class. The details of the semantics
941   are described in the association traversal operations (see 7.7.2, and 7.12.2).

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

943   `query-parameter =/ associatedrole-query-parm`
944
945   `associatedrole-query-parm = "$associatedrole=" reference-name`

946   Where:

947   • The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
948   delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)

949   • `reference-name` is the name of the reference property referencing the associated class. Note
950   that CIM property names do not contain reserved characters (see 6.3 and [DSP0004](#))

951   The `$associatedrole` query parameter shall not be repeated in a resource identifier.

952   Examples:

953   (not specified)

954   specifies no filtering on the associated role name

955   `$associatedrole=Device`

956   specifies filtering on the associated role name "Device"

6.6.3 $associationclass (association class filter)

957

958 The `$associationclass` query parameter is used to specify a filter in association traversal operations
959 that filters the result on the name of the association class. The details of the semantics are described in
960 the association traversal operations (see 7.7.2 and 7.12.2).

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

```
962   query-parameter =/ associationclass-query-parm
963
964   associationclass-query-parm = "$associationclass=" class-name
```

965 Where:

966   • The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
967   delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)

968   • `class-name` is the name of the association class (including schema prefix). Note that CIM
969   class names do not contain reserved characters (see 6.3 and DSP0004)

970 The `$associationclass` query parameter shall not be repeated in a resource identifier.

971 Examples:

972   (not specified)

973   specifies no filtering on the association class name

974   `$associationclass=ACME_SystemDevice`

975   specifies filtering on the association class name "ACME_SystemDevice"

6.6.4 $class (specify class name)

976

977 The `$class` query parameter is used to specify a class name to select matching class resources from a
978 class collection resource or instances of the named class from an instance collection resource.

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

```
980   query-parameter =/ class-query-parm
981
982   class-query-parm = "$class=" class-name
```

983 Where:

984   • The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
985   delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)

986   • `class-name` is the name of the class (including schema prefix). Note that CIM class names do
987   not contain reserved characters (see 6.3 and DSP0004)

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

989 Examples:

990   (not specified)

991   specifies no class name

992   `$class=ACME_ComputerSystem`

993   specifies class name "ACME_Computersystem"

6.6.5 $continueonerror (continue on errors within paged retrieval)

994

995 The `$continueonerror` query parameter specifies whether or not the server continues paged retrieval
996 sequences in case of errors (instead of closing them). For details about paged retrieval, see 7.3.7.

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

```
998   query-parameter =/ continueonerror-query-parm
999
1000  continueonerror-query-parm = "$continueonerror=" ( "true" / "false" )
```

1001 Where:

1002   •   The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
1003   delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)

1004 Note that the values "true" and "false" are treated case sensitively, as defined in 6.3

1005 The `$continueonerror` query parameter shall not be repeated in a resource identifier.

1006 Omitting the `$continueonerror` query parameter or specifying it with a value of "false" shall cause the
1007 server to close paged retrieval sequences in case of errors.

1008 Specifying the `$continueonerror` query parameter with a value of "true" shall cause the server to
1009 continue paged retrieval sequences in case of errors.

1010 Examples:

1011   (not specified)
1012   `$continueonerror=false`

1013   The server closes paged retrieval sequences in case of errors

1014   `$continueonerror=true`

1015   The server continues paged retrieval sequences in case of errors

6.6.6 $filter (filter instances in result)

1016

1017 The `$filter` query parameter acts as a restricting filter on the set of instances included in an instance
1018 collection.

1019 The filter language in which the value of the `$filter` parameter is to be interpreted is specified using the
1020 $filterlanguage parameter (see 6.6.7).

1021 The format of the `$filter` query parameter is defined by the following ABNF:

```
1022  query-parameter =/ filter-query-parm
1023
1024  filter-query-parm = "$filter=" [ filter-query ]
```

1025   Where:

- 1026   • The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
- 1027   delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-
- 1028   encoded).

- 1029   • `filter-query` is a filter query string that shall conform to the format of the filter language
- 1030   specified with the `$filterlanguage` parameter (or its default if not specified); if it evaluates to
- 1031   true for an instance then the instance is included, otherwise, it is not included.

- 1032   Any reserved characters that occur in the filter query string shall be considered data for
- 1033   purposes of percent-encoding (see 6.3, that is, they shall be percent-encoded).

1034   The `$filter` query parameter may be repeated in a resource identifier, see 6.6. Multiple occurrences of
1035   the `$filter` query parameter shall be combined by using logical AND on the filter query of each single
1036   parameter value.

1037   The `$filter` query parameter may be specified only in resource identifiers of instance collection
1038   resources.

1039   Omitting the `$filter` query parameter shall result in no additional restrictive filtering of instances in the
1040   instance collection.

1041   A `$filter` query parameter that is specified with no value shall result in including no instances from the
1042   instance collection.

1043   Examples (using FQL as a filter language):

1044   (not specified)

1045   no additional restrictive instance filtering takes place

1046   `$filter=`

1047   includes no instances

1048   `$filter=Type%3D%27LAN%27%20AND%20ErrorRate%3E0`

1049   specifies the FQL query string `Type='LAN' AND ErrorRate>0` and causes only instances
1050   with properties Type = "LAN" and ErrorRate > 0 to be included.

1051   The characters "=" and single quote (') in the query parameter value are percent-encoded
1052   because they are reserved characters.

1053   The blank and ">" characters in the query parameter value are percent-encoded because they
1054   are neither reserved nor unreserved characters.

1055   `$filter=Type%3D%27LAN%27&$filter=ErrorRate%3E0`

1056   specifies the same as the previous filter query; it is just split into two occurrences of the
1057   `$filter` query parameter.

1058   `$filter=Description%3D%27a%2Cb%3D0%27`

1059   specifies the FQL query string `Description='a,b=0'` and causes only instances with
1060   property Description = "a,b=0" to be included.

1061   The characters "=", comma (,) and single quote (') in the query parameter value are percent-
1062   encoded because they are reserved characters.

1063 **6.6.7 $filterlanguage (specify filter language)**

1064   The `$filterlanguage` query parameter specifies the filter language for the `$filter` parameter (see
1065   6.6.6).

1066   In this version of CIM-RS, support for the DMTF *Filter Query Language* (FQL) defined in DSP0212 is
1067   required. Other filter languages may be supported in addition.

1068   The format of this query parameter is defined by the following ABNF:

```
1069   query-parameter =/ filterlanguage-query-parm
1070
1071   filterlanguage-query-parm = "$filterlanguage=" filter-language
```

1072   Where:

1073   • The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
1074   delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-
1075   encoded).

1076   • `filter-language` specifies the filter query language, using an identifier defined by the filter
1077   language specification. The filter language is treated case-insensitively.

1078   Any reserved characters that occur in the filter language string shall be considered data for
1079   purposes of percent-encoding (see 6.3, that is, they shall be percent-encoded).

1080   DSP0212 defines the string `"DMTF:FQL"` as an identifier for FQL.

1081   The `$filterlanguage` query parameter may be specified only when the `$filter` parameter is
1082   specified.

1083   Omitting the `$filterlanguage` query parameter shall cause the filter language to default to FQL.

1084   Examples:

1085   (not specified)

1086   FQL is used by default

1087   `$filterlanguage=DMTF%3AFQL`

1088   FQL is specified explicitly. The colon ":" in the identifier string is percent-escaped because it is a
1089   reserved character.

1090 **6.6.8 $max (limit number of collection members in result)**

1091   The `$max` query parameter limits the number of members in any retrieved collections to the specified
1092   number.

1093   If there are members in excess of that maximum number, the server shall return the collection in paged
1094   mode. Note that a server may choose to return the collection in paged mode also when the specified
1095   maximum number of members is not exceeded. For details on paging of collections, see 7.3.7.

1096   The format of this query parameter is defined by the following ABNF:

```
1097    query-parameter =/ max-query-parm
1098
1099    max-query-parm = "$max=" max-members
1100
1101    max-members = nonNegativeDecimalInteger
```

1102   Where:

1103   • The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
1104   delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)

1105   • `max-members` specifies the maximum number of collection members.

1106   The `$max` query parameter shall not be repeated in a resource identifier.

1107   Omitting the `$max` query parameter indicates that there is no maximum number specified.

1108   Specifying the `$max` query parameter with a value of 0 indicates that a collection with no members shall
1109   be returned.

1110   Note that a server may choose to use paging also when the no maximum is specified.

1111   Examples:

1112   (not specified)

1113   no maximum is specified by the client for the number of members in the collection result. Note
1114   that the server may still implement a maximum, and may still use paging for the result (see
1115   7.3.7).

1116   `$max=0`

1117   number of members in the collection result is limited to no more than 0 (that is, the collection is
1118   empty).

1119   `$max=10`

1120   number of members in the collection result is limited to no more than 10.

1121   6.6.9 $pagingtimeout (specify inactivity timeout for paged retrieval)

1122   The `$pagingtimeout` query parameter specifies a duration after which a server may close a sequence
1123   of paged retrievals of subset collections if there is no retrieval activity on that sequence. This duration is
1124   referred to as *paging timeout*. For details, see 7.3.7.

1125   The format of this query parameter is defined by the following ABNF:

```
1126    query-parameter =/ pagingtimeout-query-parm
1127
1128    pagingtimeout-query-parm = "$pagingtimeout=" duration
1129
1130    duration = nonNegativeDecimalInteger
```

1131 Where:

1132 • The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
1133 delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)

1134 • `duration` is the duration of the paging timeout in seconds. A value of 0 specifies that there is
1135 no paging timeout (that is, an infinite paging timeout)

1136 The `$pagingtimeout` query parameter shall not be repeated in a resource identifier.

1137 Omitting the `$pagingtimeout` query parameter shall result in using a paging timeout that is specific to
1138 the server implementation.

1139 The allowable values for the paging timeout clients may specify with the `$pagingtimeout` query
1140 parameter are not defined at the level of the CIM-RS protocol; that is left to management instrumentation
1141 of the server.

1142 Examples:

1143 (not specified)

1144 a paging timeout specific to the server implementation is used

1145 `$pagingtimeout=0`

1146 no paging timeout is used (infinite paging timeout)

1147 `$pagingtimeout=30`

1148 a paging timeout of 30 seconds is used

6.6.10 $properties (subset properties in result)

1150 The `$properties` query parameter subsets the properties in any retrieved instance representations to
1151 only the specified set of properties. This is semantically equivalent to acting on a different resource that is
1152 a subset of the full resource.

1153 The format of this query parameter is defined by the following ABNF:

```
1154 query-parameter =/ properties-query-parm
1155
1156 properties-query-parm = "$properties=" [ property-list ]
1157
1158 property-list = property-name *( "," property-name )
```

1159 Where:

1160 • The reserved characters "$", "=" and "," in the literals of these ABNF rules shall be considered
1161 delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-
1162 encoded).

1163 • `property-name` is the name of a property in the instances. Note that CIM property names do
1164 not contain reserved characters (see 6.3 and DSP0004).

1165 The `$properties` query parameter may be repeated in a resource identifier, see 6.6. If repeated, the
1166 effective property list shall be the combined property list of all occurrences of the `$properties` query
1167 parameter.

1168 Omitting the `$properties` query parameter shall result in not excluding any properties.

1169 A `$properties` query parameter that is specified with no value shall result in including no properties in
1170 the retrieved instance representations.

1171 The order of property names specified in the query parameter is not relevant for the order of properties in
1172 the retrieved instance representations.

1173 This query parameter may be specified only in resource identifiers of instance resources or instance
1174 collection resources. If specified in resource identifiers of instance collection resources, it applies to all
1175 instances in the collection.

1176 A reference to a property that is an embedded instance or a structure shall cause all underlying properties
1177 to be included.

1178 Duplicate and invalid property names shall be ignored. Invalid property names are names of properties
1179 that are not exposed by the creation class of an instance.

1180 Examples:

1181   (not specified)

1182   no properties are excluded

1183   `$properties=`

1184   no properties are included

1185   `$properties=Name,Type`

1186   only the properties "Name" and "Type" are included

6.6.11 $qualifiers (include qualifiers in returned classes)

1188 The `$qualifiers` query parameter specifies whether or not to include qualifiers in any returned classes
1189 (see 7.10.2).

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

```
1191 query-parameter =/ qualifiers-query-parm
1192
1193 qualifiers-query-parm = "$qualifiers=" ( "true" / "false" )
```

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, that is, they shall not be percent-encoded)

1197 Note that the values "true" and "false" are treated case sensitively, as defined in 6.3

1198 The `$qualifiers` query parameter shall not be repeated in a resource identifier.

1199 Omitting the `$qualifiers` query parameter or specifying it with a value of "false" shall cause the server
1200 to not include qualifiers in any returned classes.

1201 Specifying the `$qualifiers` query parameter with a value of "true" shall cause the server to include
1202 qualifiers in any returned classes.

1203   Examples:
1204   ```
        (not specified)
1205 $qualifiers=false
        ```

1206            No qualifiers are included in any returned classes.
1207    ```
        $qualifiers=true
        ```

1208   Qualifiers are included in any returned classes.

6.6.12 $sourcerole (source role filter)

1210   The `$sourcerole` query parameter is used to specify a filter in association traversal operations that
1211   filters the result on the role name for the source class; that is, the name of the reference property in the
1212   traversed association that references the source class. The details of the semantics are described in the
1213   association traversal operations (see 7.7.2, 7.8.2, 7.12.2, and 7.13.2).

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

1215   ```
        query-parameter =/ sourcerole-query-parm
1216
1217   sourcerole-query-parm = "$sourcerole=" reference-name
        ```

1218    Where:

1219        •   The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
1220            delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)

1221        •   `reference-name` is the name of the reference property referencing the source class. Note that
1222            CIM property names do not contain reserved characters (see 6.3 and DSP0004)

1223    The `$sourcerole` query parameter shall not be repeated in a resource identifier.

1224    Examples:

1225        (not specified)

1226            specifies no filtering on the source role name

1227    ```
        $sourcerole=System
        ```

1228   specifies filtering on the source role name "System"

6.6.13 $subclasses (include subclasses in class enumeration result)

1230   The `$subclasses` query parameter specifies whether or not the (direct and indirect) subclasses of a
1231   class are included in the result of a class enumeration operation (see 7.11.4).

1232   The format of this query parameter is defined by the following ABNF:

1233   ```
        query-parameter =/ subclasses-query-parm
1234
1235   subclasses-query-parm = "$subclasses=" ("true" / "false")]
        ```

1236    Where:

1237        •   The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered
1238            delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)

1239    Note that the values "true" and "false" are treated case sensitively, as defined in 6.3

1240 The `$subclasses` query parameter shall not be repeated in a resource identifier.

1241 Omitting the `$subclasses` query parameter or specifying it with a value of "false" shall cause the server
1242 to not include subclasses in the result.

1243 Specifying the `$subclasses` query parameter with a value of "true" shall cause the server to include
1244 subclasses in the result.

1245 Examples:
```
1246   (not specified)
1247 $subclasses=false
```

1248       No subclasses are included into the class collection.
```
1249 $subclasses=true
```

1250       Subclasses are included into the class collection.

# 1251 7    Resources, operations and payload elements

1252 This clause defines the types of resources used in the CIM-RS protocol, the HTTP methods (operations)
1253 on these resources, and the payload elements used in the HTTP protocol.

## 1254 7.1    Overview

1255 Table 2 shows an overview of all types of resources used in the CIM-RS protocol. A resource in the CIM-
1256 RS protocol is anything that can be the target of an HTTP method. Except for the listener indication
1257 delivery resource, these resources are located in a server.

1258                                    **Table 2 – Resource types in CIM-RS**

| Resource Type | Represents |
|---|---|
| Instance | a CIM instance, representing a modeled object in the managed environment |
| Instance collection | a collection of instances of a particular class |
| Instance associator collection | a collection of instances associated to a particular instance |
| Instance reference collection | a collection of association instances referencing a particular instance |
| Instance collection page | a page of a paged instance collection |
| Class | a CIM class, representing the type of a CIM instance |
| Class collection | a collection of classes (top-level classes in a namespace, or subclasses of a class) |
| Class associator collection | a collection of classes associated to a particular class |
| Class reference collection | a collection of association classes referencing a particular class |
| Qualifier type | a CIM qualifier type, representing the declaration of a metadata item |
| Qualifier type collection | a collection of qualifier types in a particular namespace |
| Listener indication delivery | a resource within a listener that is used to deliver indications to |

1259 A combination of a particular HTTP method on a particular type of resource is termed an "operation" in
1260 this document.

1261  Table 3 shows all operations used in the CIM-RS protocol, identified by their HTTP method and target
1262  resource type.

1263                                    **Table 3 – CIM-RS operations**

| HTTP Method | Target Resource Type | Purpose | Corresponding Generic Operation | Description |
|---|---|---|---|---|
| GET | Instance | Retrieve an instance | GetInstance | see 7.5.5 |
| PUT | Instance | Update an instance | ModifyInstance | see 7.5.6 |
| DELETE | Instance | Delete an instance | DeleteInstance | see 7.5.7 |
| POST | Instance | Invoke a method on an instance | InvokeMethod, InvokeStaticMethod on instance | see 7.5.8 |
| POST | Instance collection | Create an instance | CreateInstance | see 7.6.3 |
| GET | Instance collection | Enumerate instances of a class | OpenEnumerateInstances | see 7.6.4 |
| GET | Instance associator collection | Retrieve associated instances | OpenAssociatorInstances | see 7.7.2 |
| GET | Instance reference collection | Retrieve referencing instances | OpenReferenceInstances | see 7.8.2 |
| GET | Instance collection page | Retrieve instance collection page | PullInstancesWithPath | see 7.9.2 |
| DELETE | Instance collection page | Close paged instance collection | CloseEnumeration | see 7.9.3 |
| GET | Class | Retrieve a class | GetClass | see 7.10.3 |
| PUT | Class | Update a class | ModifyClass | see 7.10.4 |
| DELETE | Class | Delete a class | DeleteClass | see 7.10.5 |
| POST | Class | Invoke a method on a class | InvokeStaticMethod on class | see 7.10.6 |
| POST | Class collection | Create a class | CreateClass | see 7.11.3 |
| GET | Class collection | Enumerate classes in a namespace | EnumerateClasses | see 7.11.4 |
| GET | Class associator collection | Retrieve associated classes | AssociatorClasses | see 7.12.2 |
| GET | Class reference collection | Retrieve referencing classes | ReferenceClasses | see 7.13.2 |
| GET | Qualifier type | Retrieve a qualifier type | GetQualifierType | see 7.14.3 |
| PUT | Qualifier type | Update a qualifier type | ModifyQualifierType | see 7.14.4 |
| DELETE | Qualifier type | Delete a qualifier type | DeleteQualifierType | see 7.14.5 |
| POST | Qualifier type collection | Create a qualifier type | CreateQualifierType | see 7.15.3 |
| GET | Qualifier type collection | Enumerate qualifier types in a namespace | EnumerateQualifierTypes | see 7.15.4 |
| POST | Listener indication delivery | Deliver an indication | DeliverIndication | see 7.16.3 |

1264 Most of the operations used in the CIM-RS protocol have protocol payload data either in the request
1265 message, or in the response message, or both. These payload elements often correspond directly to
1266 resources, but not always. This document defines these payload element*s* in a normative but abstract
1267 way. CIM-RS payload representation specifications define how each of these payload elements is
1268 represented, for details see clause 9. The payload elements have a name for ease of referencing
1269 between documents, as shown in the first column of Table 4.

1270 Table 4 shows all payload elements used in the CIM-RS protocol.

1271                                    **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 7.5.2 |
| InstanceCollection | A list of representations of instance resources | See 7.6.2 |
| Class | Representation of a class resource; that is, a class declaration | See 7.10.2 |
| ClassCollection | A list of representations of class resources | See 7.11.2 |
| QualifierType | Representation of a qualifier type | See 7.14.2 |
| QualifierTypeCollection | A list of representations of qualifier types | See 7.15.2 |
| MethodRequest | The data describing a method invocation request, including input parameters | See 7.5.3 |
| MethodResponse | The data describing a method invocation response, including its return value and output parameters | See 7.5.4 |
| IndicationDeliveryRequest | The data describing a request to deliver an indication to a listener | See 7.16.2 |
| ErrorResponse | The data describing an error response to any request | See 7.3.5 |

## 1272 7.2    Description conventions

### 1273 7.2.1    Data types used in payload element definitions

1274 This subclause defines the data types used in the definition of the attributes of payload elements. In order
1275 to distinguish these kinds of data types from CIM data types, they are termed "payload data types".
1276 Payload data types are used as a description mechanism for this document and for any payload
1277 representation specifications.

1278 The representation of values of payload data types is defined in payload representation specifications; for
1279 details see clause 9.

1280 The payload data types used in CIM-RS are defined in Table 5. This definition allows payload
1281 representations to include or omit type information in values of properties, method parameters and
1282 method return values.

1283                                    **Table 5 – CIM-RS payload data types**

| Payload data type | Description |
|---|---|
| Boolean | a boolean value, or Null |
| String | a string of UCS characters, or Null |
| Integer | an integer value, or Null |

| Payload data type | Description |
|---|---|
| URI | a CIM-RS resource identifier, in the format defined in 6.1 |
| Value | A value of a CIM type, or Null. The value is represented as defined by the payload representation specification. |
| ElementValue | a complex type for representing the value of a typed CIM element (such as properties, method parameters or method return values), containing the following child attributes: |

| Attribute | Payload data type | Requirement | Description |
|---|---|---|---|
| Name | String | Mandatory | name of the element |
| Array | Boolean | Conditional | specifies whether the element is an array. Condition: Type information is included and the element is an array. Default if not specified: False. |
| Arraysize | Integer, or None | Conditional | specifies the size of the fixed-size array. Condition: Type information is included and the array is an array. A value of NULL indicates that the array is variable-sized. Default if not specified: NULL. |
| Type | String | Conditional | CIM-RS type name of the element, as defined in Table 6. Condition: Type information is included. |
| Classname | String | Conditional | class name related to the CIM-RS type name of the element, as defined in Table 6. Condition: Type information is included and the CIM data type requires a class name to be specified, see Table 6. Default if not specified: Not applicable. |
| Value | Value | Mandatory | value of the element |

| Payload data type | Description |
|---|---|
| QualifierValue | a complex type for CIM qualifier values, containing the following child attributes: |

| Attribute | Payload data type | Requirement | Description |
|---|---|---|---|
| name | String | Mandatory | name of the qualifier. |
| array | Boolean | Conditional | specifies whether the qualifier is an array. Condition: The element is an array. Default if not specified: False. |
| type | String | Mandatory | CIM-RS type name of the qualifier, as defined in Table 6. |
| value | Value | Mandatory | value of the qualifier. |

| Payload data type | Description | | | |
|---|---|---|---|---|
| ElementDefinition | a complex type for the definition of an element (property, reference or method parameter), containing the following child attributes: | | | |

| Attribute | Payload data type | Requirement | Description |
|---|---|---|---|
| name | String | Mandatory | name of the represented element |
| qualifiers | QualifierValue [ ] | Conditional | the CIM qualifiers defined on the element.<br>Condition: There are such qualifiers. |
| array | Boolean | Conditional | specifies whether the element is an array.<br>Condition: The element is an array.<br>Default if not specified: False. |
| arraysize | Integer, or None | Conditional | specifies the size of the fixed-size array.<br>Condition: The array is an array. A value of NULL indicates that the array is variable-sized.<br>Default if not specified: NULL. |
| type | String | Mandatory | CIM-RS type name of the element, as defined in Table 6. |
| classname | String | Conditional | class name related to the CIM-RS type name of the element, as defined in Table 6.<br>Condition: The CIM data type requires a class name to be specified, see Table 6.<br>Default if not specified: Not applicable. |
| defaultvalue | Value | Conditional | default value for the property.<br>Condition: The represented element is a property and the property has a non-Null default value.<br>Default if not specified: Null. |

| Payload data type | Description | | | |
|---|---|---|---|---|
| MethodDefinition | a complex type for the definition of a method (including its return value), containing the following child attributes: | | | |

| Attribute | Payload data type | Requirement | Description |
|---|---|---|---|
| name | String | Mandatory | name of the method (without any parenthesis or method parameters) |
| qualifiers | QualifierValue [ ] | Conditional | the CIM qualifiers defined on the method. Condition: There are such qualifiers. |
| classname | String | Conditional | class name related to the CIM-RS type name of the method return value, as defined in Table 6. Condition: CIM data type requires class name to be specified, see Table 6. Default if not specified: Not applicable. |
| type | String | Mandatory | CIM-RS type name of the method return value, as defined in Table 6. Note that a method cannot return a reference type in CIM. |
| parameters | ElementDefinition [ ] | Conditional | definition of each method parameter. Condition: There are such parameters. |

| Payload data type | Description |
|---|---|
| Instance | an Instance payload element, as defined in 7.5.2 |
| Class | a Class payload element, as defined in 7.10.2 |
| QualifierType | a QualifierType payload element, as defined in 7.14.2 |

1284 The CIM data type specified in the "type" child element of the ElementValue type allows infrastructure
1285 components to represent element values in programming environments using strong types for the CIM
1286 data types. This is expected to be used for WBEM client implementations as model-neutral client libraries.

### 1287  7.2.2   Data type names

1288 The type names to be used for the "type" attribute of various payload elements, and related other
1289 attributes are defined in Table 6. In most cases, the CIM-RS type names correspond 1:1 to CIM type
1290 names. However, in the areas of embedded objects, CIM-RS has specific type names instead of using
1291 the string type as in CIM.

1292                                 **Table 6 – Names of CIM-RS data types**

| CIM data type | CIM-RS type name | Additional rules |
|---|---|---|
| boolean | boolean | |
| string | string | |
| char16 | char16 | |
| string, with OctetString qualifier | string | |
| uint8[], with OctetString qualifier | uint8 | The "array" attribute shall be True |

| CIM data type | CIM-RS type name | Additional rules |
|---|---|---|
| string with EmbeddedInstance(<classname>) qualifier | instance | The "classname" attribute shall specify the creation class of the embedded instance |
| string with EmbeddedObject qualifier containing an embedded instance | instance | The "classname" attribute shall specify the creation class of the embedded instance |
| string with EmbeddedObject qualifier containing an embedded class | class | The "classname" attribute shall specify the embedded class |
| datetime | datetime | |
| uint8,16,32,64 | uint8,16,32,64 | |
| sint8,16,32,64 | sint8,16,32,64 | |
| real32,64 | real32,64 | |
| <classname> ref | reference | The "classname" attribute shall specify the class declared in the reference (<classname>) |
| string with Reference(<classname>) qualifier | reference | The "classname" attribute shall specify the creation class of the referenced instance |
| array of any CIM type | <type name of array elements> | The "array" attribute shall be True |

### 7.2.3 Requirement levels used in payload element definitions

1293

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

1294
1295

| | |
|---|---|
| **Mandatory** | The attribute shall be included in the payload element. |
| **Conditional** | The attribute shall be included in the payload element if the condition is met. If the condition is not met, the attribute may be included in the payload element at the discretion of the implementation. |
| **ConditionalExclusive** | The attribute shall be included in the payload element if the condition is met. If the condition is not met, the attribute shall not be included in the payload element. |
| **Optional** | The attribute may be included in the payload element at the discretion of the implementation. |

1296

1297
1298
1299

1300
1301
1302

1303
1304

### 7.2.4 Requirement levels used in operation definitions

1305

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

1306

| | |
|---|---|
| **Mandatory** | The operation shall be implemented by the server or listener. |
| **Optional** | The operation may be implemented, at the discretion of the server or listener implementation. |
| **Class-specific** | The requirement to implement the operation by the server is specific to the use of a class in a model (for example, as defined in management profiles). |

1307

1308
1309

1310
1311
1312

1313 **7.2.5 Description format for operations**

1314 The definition of operations in the following subclauses uses the following description fields:

1315 **Purpose:** A brief description of the purpose of the operation.

1316 **HTTP method:** The name of the HTTP method used to perform the operation (for
1317 example, GET, PUT, POST, DELETE).

1318 **Target resource:** The type of resource that is identified as the target of the HTTP method,
1319 by means of the Request-URI field (see RFC2616) and Host header
1320 field.

1321 **Query parameters:** The names of any query parameters that may be specified in the
1322 resource identifier. Other query parameters shall not be specified by the
1323 requester. If other query parameters are specified by the requester, they
1324 shall be ignored by the responder, in order to provide for future
1325 extensibility.

1326 **Request headers:** The names of any header fields that may be specified in the request
1327 message. Other request headers shall not be specified by the requester.
1328 If other query request headers are specified by the requester, they shall
1329 be ignored by the responder, in order to provide for future extensibility.

1330 **Request payload:** The name of the payload element that shall be used in the entity body of
1331 the request message. "None" means the entity body shall be empty.

1332 **Response headers:** The names of any header fields that may be specified in the response
1333 message, separately for the success and failure cases. Other response
1334 headers shall not be specified by the responder. If other query request
1335 headers are specified by the responder, they shall be ignored by the
1336 requester, in order to provide for future extensibility.

1337 **Response payload:** The name of the payload element that shall be used in the entity body of
1338 the response message, separately for the success and failure cases.
1339 "None" means the entity body shall be empty.

1340 **Requirement:** The requirement level to implement the operation, as defined in 7.2.4.

1341 **Description:** A normative definition of the behavior of the operation, in addition to the
1342 normative definitions stated in this subclause. Normative requirements in
1343 this subclause are sometimes directed to the provider of the operation,
1344 and sometimes to its consumer.

1345 **Example HTTP conversation:** An example HTTP request and HTTP response. The examples are
1346 informative and use the CIM-RS payload representation in JSON as
1347 defined in DSP0211. They do not show all cases of using query
1348 parameters or all cases of including or not including type information (a
1349 concept supported by DSP0211) In case of differences between these
1350 examples and DSP0211, the latter is authoritative.

1351 **7.3 Common behaviors for all operations**

1352 **7.3.1 Content negotiation**

1353 In order to determine the type of CIM-RS payload representation to be used, WBEM clients, servers, and
1354 listeners shall support server-driven content negotiation as defined in RFC2616, based on the Accept

---

1355    request-header (defined in RFC2616 and in 8.4.1), and the Content-Type response header field (defined
1356    in RFC2616 and in 8.4.2).

1357    Requirements for the media types used in these header fields are defined in 9.1.

1358    The supported types of CIM-RS payload representations cannot be discovered at the level of the CIM-RS
1359    protocol; that is left to the management instrumentation of a server.

### 7.3.2   Caching of responses

1360

1361    Caching of responses from servers and listeners is described in RFC2616. This document does not
1362    define any additional constraints or restrictions on caching.

1363    Note that any use of the HTTP GET method in the CIM-RS protocol is safe and idempotent, and that any
1364    use of the HTTP PUT method in the CIM-RS protocol is idempotent.

### 7.3.3   Success and failure

1365

1366    Operations performed within the CIM-RS protocol shall either succeed or fail. There is no concept of
1367    "partial success" in the CIM-RS protocol.

1368    If an operation succeeds, it shall return its output data to the operation requester and shall not include any
1369    errors.

1370    If an operation fails, it shall return an error to the operation requester (see 7.3.5) and no other output data.

1371    For example, if an instance collection retrieval operation were able to return some, but not all, instances
1372    successfully, then the operation fails without returning any instances.

1373    When using paged retrieval, each retrieval operation within a paged retrieval stream is considered a
1374    separate operation w.r.t. success and failure.

1375    Servers may implement a streaming approach for paged retrieval, by sending returned instances back to
1376    the client while they are still being built up, in order to lower the amount of memory consumed by the
1377    server. Such a server may encounter errors after some portion of the response has already been sent
1378    back to the client. Consistent with the approach for success and failure described in this subclause, the
1379    server can finish the current response with success, returning only good instances in that response (i.e.
1380    before the error happened), and keeping the error until the next page is requested by the client. That next
1381    page will then return no instances, but an error (see 7.3.5).

### 7.3.4   Errors

1382

1383    Errors at the CIM-RS protocol level are returned as HTTP status codes. The definition of HTTP status
1384    codes defined in RFC2616 is the basis for each operation, and the operation descriptions in this
1385    document specify any additional constraints on the use of HTTP status codes.

1386    Table 7 lists HTTP status codes that may be returned by any HTTP method defined in this document.

1387                                **Table 7 – HTTP status codes for any HTTP method**

| HTTP status code | HTTP status text | Generic operations error ID | Generic operations error title |
|---|---|---|---|
| 401 | Unauthorized | WIPG0201 | Access denied |
| 503 | Service Unavailable | WIPG0236 | WBEM server is shutting down |
| 503 | Service Unavailable | WIPG0240 | WBEM server limits are exceeded |

| HTTP status code | HTTP status text | Generic operations error ID | Generic operations error title |
|---|---|---|---|
| 408 | Request Timeout | WIPG0243 | Timeout |
| 405 | Method Not Allowed | WIPG0227 | Other failure |
| 406 | Not Acceptable | WIPG0227 | Other failure |
| 411 | Length Required | WIPG0227 | Other failure |
| 413 | Request Entity Too Large | WIPG0227 | Other failure |
| 414 | Request-URI Too Long | WIPG0227 | Other failure |
| 415 | Unsupported Media Type | WIPG0227 | Other failure |
| 429 | Too Many Requests | WIPG0227 | Other failure |
| 431 | Request Header Fields Too Large | WIPG0227 | Other failure |
| 500 | Internal Server Error | WIPG0227 | Other failure |
| 503 | Service Unavailable | WIPG0227 | Other failure |
| 505 | HTTP Version Not Supported | WIPG0227 | Other failure |

1388  Extended error information is returned as an ErrorResponse payload element (see 7.3.5) in the entity
1389  body. For details about its usage, see the operation descriptions in clause 7.

1390  **7.3.5  ErrorResponse payload element**

1391  An ErrorResponse payload element represents the data used in an error response to any request.

1392  An ErrorResponse payload element shall have the following attributes:

1393                        **Table 8 – Attributes of an ErrorResponse payload element**

| Attribute name | Payload data type | 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 | Mandatory | CIM status code |
| statusdescription | String | Mandatory | CIM status description |
| errors | Instance [ ] | Conditional | order-preserving list of zero or more embedded instances of class CIM_Error defined in the CIM Schema published by DMTF, each specifying an error message. Condition: There are such instances. |

1394    **Example HTTP error response of a failed GET instance (using JSON as defined in DSP0211):**

1395    Response (if type information is included):

```
1396    HTTP/1.1 404 Not Found
1397    Date: Thu, 30 Oct 2014 15:03:00 GMT
1398    Content-Length: XXX
1399    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
1400    X-CIMRS-Version: 2.0.1
1401
1402    {
1403      "kind": "errorresponse",
1404      "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
1405    ys11",
1406      "httpmethod": "GET",
1407      "statuscode": 6,
1408      "statusdescription": "WIPG0213: CIM instance ACME_VirtualSystem.InstanceID=\"node
1409    47:sys11\" does not exist in CIM namespace root/cimv2.",
1410      "errors": [
1411        {
1412          "kind": "instance",
1413          // self is omitted for embedded instances
1414          // namespace is omitted for embedded instances
1415          "classname": "CIM_Error",
1416          "properties": {
1417            "ErrorType": {
1418              "type": "uint16",
1419              "value": 4},
1420            "ErrorSource": {
1421              "type": "string",
1422              "value": "root/cimv2:ACME_VirtualSystem.InstanceID=\"node47:sys11\""},
1423            "ErrorSourceFormat": {
1424              "type": "uint16",
1425              "value": 2},
1426            "Message": {
1427              "type": "string",
1428              "value": "WIPG0213: CIM instance ACME_VirtualSystem.InstanceID=\"node47:s
1429    ys11\" does not exist in CIM namespace root/cimv2."},
1430            "MessageArguments": {
1431              "type": "string",
1432              "array": true,
1433              // arraysize is omitted
1434              "value": [
1435                "ACME_VirtualSystem.InstanceID=\"node47:sys11\"",
1436                "root/cimv2",
1437                "GetInstance",
1438                null,
1439                "root/cimv2:ACME_VirtualSystem.InstanceID=\"node47:sys11\""
1440              ]},
1441            "MessageID": ": {
```

```
1442              "type": "string",
1443              "value": "WIPG0213"},
1444          "OwningEntity": {
1445              "type": "string",
1446              "value": "DMTF"}
1447        }
1448      }
1449    ]
1450  }
```

7.3.6 Consistency model

1452 The operations of the CIM-RS protocol shall conform to the consistency model defined in <u>DSP0223</u>.

7.3.7 Paging of instance collections

1454 Client and servers shall support the *paging of instance collections* returned to clients as described in this
1455 subclause, for the operations listed in Table 9.

1456 When an instance collection is being retrieved by a client, the server may choose to use paging for the
1457 collection, at the server's discretion.

1458 If the server does not use paging for an instance collection, the "next" attribute of the returned
1459 representation of the collection shall be omitted.

1460 If the server uses paging for an instance collection, the "next" attribute of the returned representation of
1461 the collection shall reference a instance collection page resource (see 7.9) that contains the next subset
1462 of collection members (= page). That next subset collection may again contain only a subset of the
1463 remaining members, and so forth. The last subset collection has no "next" attribute, indicating that it is the
1464 last one of the sequence of subset collections.

1465 As a result, any returned representation of a collection subset is self-describing w.r.t. whether it contains
1466 the last (or possibly only) set of members, or other subsets are following; and the subdivision of the
1467 complete set of collection members into subset collections always happens at a granularity of complete
1468 instances so that these instances are never broken apart to be returned in separate subset collections.

1469 Table 9 lists the operations that may open paged instance collections:

1470 **Table 9 – Operations that may open paged instance collections**

HTTP Method	Target Resource Type	Retrieved Resource Representation	Description
GET	Instance collection	instance collection	see 7.6.4
GET	Instance associator collection	instance collection	see 7.7.2
GET	Instance reference collection	instance collection	see 7.8.2

1471 Table 10 lists other operations related to paged instance collections:

1472   **Table 10 – Other operations related to paged instance collections**

HTTP Method	Target Resource Type	Retrieved Resource Representation	Description
GET	Instance collection page	instance collection	see 7.9.2
DELETE	Instance collection page		see 7.9.3

1473 Clients may use the $max query parameter (see 6.6.8) to limit the number of instances in each returned
1474 instance collection page.

1475 Each returned instance collection page shall contain any number of instances between zero and the
1476 maximum specified with the $max query parameter (if specified). The number of instances in a instance
1477 collection page may vary over the course of retrieving the entire collection. As a result, the number of
1478 instances in a subset collection is not a safe indicator for a client that there are remaining instances; only
1479 the presence of the "next" attribute is a safe indicator for that.

1480 The resource identifiers of any two instance collection page resources that belong to different open paged
1481 instance collections shall be distinct. The resource identifiers of any two instance collection page
1482 resources that belong to the same open paged instance collection do not need to be distinct. Servers
1483 have these options for representing the retrieval state of a paged instance collection:

1484   • By maintaining the entire retrieval state in a value that is encoded in the resource identifier of
1485   the page. This will cause the server to be stateless w.r.t. the retrieval state. In this case, the
1486   resource identifiers of different pages within the same paged instance collection will be distinct.

1487   • By maintaining the retrieval state within the server and referencing that state using a value that
1488   is encoded in the resource identifier of the page. In this case, the resource identifiers of different
1489   pages within the same paged instance collection typically will be the same.

1490 Servers should implement ceasing of instance collection page resources. If a server implements ceasing
1491 of instance collection page resources, any successfully retrieved collection page (other than the first one)
1492 shall cause its previous instance collection page resource to cease existence, and subsequent requests
1493 to retrieve such a ceased instance collection page resource shall be rejected with HTTP status code 404
1494 "Not Found". Note that ceasing of instance collection page resources can only be implemented if the
1495 resource identifiers of different pages within the same open paged instance collection are distinct.

1496 Separate retrieval requests for the (entire) collection resource shall be treated independently by the
1497 server (regardless of whether these requests come from the same or different clients, and regardless of
1498 whether a request is a repetition of an earlier request). As a result, each successful retrieval request of
1499 the entire collection opens a new sequence of paged retrievals for the remaining instance collection page
1500 resources.

1501 Clients and servers may support the "continue on error" feature (see 7.4.1). Clients that support the
1502 "continue on error" feature may request continuation on error for paged retrievals by specifying the
1503 $continueonerror query parameter (see 6.6.5). If a retrieval request results in an error, and the client
1504 has requested continuation on error, and the server supports the "continue on error" feature, the server
1505 shall not close the sequence of retrievals. Otherwise, the server shall close the sequence of retrievals, if a
1506 retrieval request results in an error. For details on this behavior, see the description of "continuation on
1507 error" of pulled enumerations in DSP0223.

1508 Servers should close a sequence of paged retrievals after some time of inactivity on that sequence, even
1509 if the client has not retrieved the sequence exhaustively. Clients may use the $pagingtimeout query
1510 parameter (see 6.6.9) to specify the minimum duration the server is obliged to keep a sequence of paged
1511 subset collections open after retrieval of a subset collection. If the $pagingtimeout query parameter is
1512 not specified, the server may use any timeout. For details on this behavior, see the description of
1513 "operation timeout" of pulled enumerations in DSP0223. Clients may close a sequence of paged retrievals
1514 using DELETE on the instance collection page resource (see 7.9.3).

1515 The concept of paging collections as described in this subclause is consistent with pulled enumerations
1516 as defined in DSP0223, so that it fits easily with servers that support the semantics of pulled
1517 enumerations in their implementation.

1518 Servers that support pulled enumerations in their implementation can achieve to be entirely stateless
1519 w.r.t. paged instance collections, by maintaining the entire state data of the paging progress in the
1520 enumeration context value, and by representing the enumeration context value in the resource identifiers
1521 of instance collection page resources. Binary data in an enumeration context value can for example be
1522 represented using a base64url encoding (see RFC4648), typically without any "=" padding characters at
1523 the end.

1524 For more details on pulled enumerations and the concept of enumeration context values, see DSP0223.

1525 NOTE   The use of HTTP range requests as defined in RFC2616 has been considered and dismissed, because the
1526 semantics of an ordered sequence of items that can be accessed by item number cannot be provided by
1527 implementations that support the opaque server-defined enumeration context values mandated by DSP0223.

7.4   Optional features of the CIM-RS protocol

1529 This subclause defines optional features for the implementation of the CIM-RS protocol.

7.4.1   "Continue on error" feature

1531 Implementation of the "continue on error" feature in servers provides clients with the possibility to request
1532 continuation of a sequence of paged retrievals in case of error. For details on paged retrieval, see 7.3.7.

1533 Implementation of the "continue on error" feature is optional for clients and servers, independently.

7.5   Instance resource

1535 An instance resource represents a managed object in the managed environment.

1536 Because CIM-RS is model-neutral, it defines how instances are exposed as instance resources. A model
1537 defines how managed objects are modeled as instances, by defining classes. In combination, the CIM-RS
1538 protocol and the model define how managed objects are represented as REST resources. For details,
1539 see 5.4.

7.5.1   Resource identifier

1541 Instance resources shall have a resource identifier whose path component (that is, the `path-absolute`
1542 ABNF rule in 6.1) matches ABNF rule `instance-path-absolute`:

```
instance-path-absolute = "/" nsname "/classes/" classname "/instances/" keys

keys = key *("," key)

key = keyname "=" keyvalue
```

1548 Where:

1549   • `nsname` is the namespace name, in its original lexical case, percent-encoded as defined in 6.3.
1550   The reserved character "/" in namespace names shall be considered data for purposes of
1551   percent-encoding (that is, it shall be percent-encoded); otherwise, namespace names do not
1552   contain reserved characters.

1553   • `classname` is the class name, in its original lexical case, percent-encoded as defined in 6.3.
1554   Note that CIM class names do not contain reserved characters (see 6.3 and DSP0004).

1555	• `keyname` is the key property name, in its original lexical case, percent-encoded as defined in
1556	6.3. Note that CIM property names do not contain reserved characters (see 6.3 and DSP0004).

1557	• `keyvalue` is the key property value. The character sequence used for this resource identifier
1558	component shall be the string representation of the CIM typed value as defined in DSP0004,
1559	with any reserved characters considered to be data (see 6.3, that is, they shall be percent-
1560	encoded).

1561 Examples:

```
1562    /root%2Fcimv2/classes/ACME_Fan/instances/InstanceID=node47%3Asys11%3Afan7
1563    /root%2Fcimv2/classes/ACME_ComputerSystem/instances/System=node47,Name=sys11
```

1564 7.5.2 Instance payload element

1565 An Instance payload element is the representation of an instance resource (and thus, of a managed
1566 object in the managed environment) in the protocol.

1567 Unless otherwise constrained, an Instance payload element shall have the attributes defined in Table 11.

1568   **Table 11 – Attributes of an Instance payload element**

Attribute name	Payload data type	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value `"instance"`
self	URI	Conditional	resource identifier of the represented instance. Condition: The instance is addressable; that is, not an embedded instance. Default if not specified: Not applicable.
namespace	String	Conditional	namespace name of the represented instance. Condition: The instance is addressable; that is, not an embedded instance. Default if not specified: Not applicable.
classname	String	Mandatory	class name of the creation class of the represented instance
properties	ElementValue [ ]	Conditional	unordered list of properties (see 7.2.1), representing all or a subset of the properties of the instance resource. Condition: The payload element includes properties.

1569 7.5.3 MethodRequest payload element

1570 A MethodRequest payload element is the representation of a request to invoke a method in the protocol.
1571 This payload element is used for invocation of methods on instances (see 7.5.8) as well as classes (see
1572 7.10.6).

1573 Unless otherwise constrained, a MethodRequest payload element shall have the attributes defined in
1574 Table 12.

1575 **Table 12 – Attributes of a MethodRequest payload element**

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

1576 **7.5.4   MethodResponse payload element**

1577 A MethodResponse payload element is the representation of the response of a method invocation in the
1578 protocol. This payload element is used for invocation of methods on instances (see 7.5.8) as well as
1579 classes (see 7.10.6).

1580 Unless otherwise constrained, a MethodResponse payload element shall have the attributes defined in
1581 Table 13.

1582 **Table 13 – Attributes of a MethodResponse payload element**

Attribute name	Payload data type	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value `"methodresponse"`
self	URI	Mandatory	resource identifier of the target resource (instance or class)
methodname	String	Mandatory	method name (without any parenthesis or method parameters)
returnvalue	ElementValue	Mandatory	method return value. Because return values of methods do not have a name, payload specifications need to clarify how the "name" child attribute is set.
parameters	ElementValue [ ]	Conditional	unordered list of method output parameters. Condition: The payload element includes method output parameters.

1583 **7.5.5   GET (retrieve an instance)**

1584 **Purpose:**   Retrieve an instance

1585 **HTTP method:**   GET

1586 **Target resource:**   Instance (see 7.5.1)

1587 **Query parameters:**   `$properties`

1588 **Request headers:**   Host, Accept, X-CIMRS-Version

1589 **Request payload:**   None

1590   **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

1591   **Response payload (success):** Instance (see 7.5.2)

1592   **Response headers (failure):**   Date, Content-Length, Content-Type, X-CIMRS-Version

1593   **Response payload (failure):**   ErrorResponse (see 7.3.5)

1594   **Requirement:**   Class-specific

1595   **Description:**

1596   The HTTP GET method on an instance resource retrieves a representation of the specified instance.

1597   For details on the effects of the query parameters on the returned Instance payload element, see the
1598   descriptions of these query parameters in 6.6.

1599   On success, one of the following HTTP status codes shall be returned:

1600   •   200 "OK": The entity body shall contain an Instance payload element representing the
1601   specified instance (see 7.5.2).

1602   •   304 "Not Modified": The validators matched on a conditional request; the entity body shall
1603   be empty. This status code can only occur if the server supports conditional requests and
1604   the client has requested a conditional request. For details on conditional requests, see
1605   subclause 9.3 in RFC2616.

1606   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
1607   the HTTP status codes in Table 14 or Table 7 shall be returned.

1608   **Table 14 – HTTP status codes for failing GET (retrieve an instance)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value
404	Not Found	WIPG0214	Class not found
501	Not Implemented	WIPG0228	Operation not supported by class implementation
404	Not Found	WIPG0213	Instance not found

1609 **Example HTTP conversation (using JSON as defined in DSP0211):**

1610 Request (if type information is accepted to be included in the response):

```
1611  GET /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11 HT
1612  TP/1.1
1613  Host: server.acme.com:5988
1614  Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
1615  X-CIMRS-Version: 2.0.0
```

1616 Response (if type information is included):

```
1617  HTTP/1.1 200 OK
1618  Date: Thu, 30 Oct 2014 15:03:00 GMT
1619  Content-Length: XXX
1620  Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
1621  X-CIMRS-Version: 2.0.1
1622
1623  {
1624    "kind": "instance",
1625    "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
1626  ys11",
1627    "namespace": "root/cimv2",
1628    "classname": "ACME_VirtualSystem",
1629    "properties": {
1630      "InstanceID": {
1631        "type": "string",
1632        "value": "node47:sys11" },
1633      "ElementName": {
1634        "type": "string",
1635        "value": "Virtual system 11 on node 07" },
1636      "Caption": {
1637        "type": "string",
1638        "value": "Virtual system 11 on node 07" },
1639      . . .
1640    }
1641  }
```

1642 **7.5.6   PUT (update an instance)**

1643 **Purpose:**   Update an instance (partially or fully)

1644 **HTTP method:**   PUT

1645 **Target resource:**   Instance (see 7.5.1)

1646 **Query parameters:** $properties

1647 **Request headers:**   Host, Accept, Content-Length, Content-Type, X-CIMRS-Version

1648 **Request payload:**   Instance (see 7.5.2)

1649 **Response headers (success):** Date, X-CIMRS-Version

1650 **Response payload (success):** None

1651 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

1652 **Response payload (failure):** ErrorResponse (see 7.3.5)

1653 **Requirement:** Class-specific

1654 **Description:**

1655 The HTTP PUT method on an instance resource updates some or all property values of the specified
1656 instance resource.

1657 Partial update of an instance is achieved by specifying the desired subset of properties in the
1658 resource identifier using the $properties query parameter (see 6.6.10). Because query
1659 parameters are part of the address of a resource (see RFC2616), this approach performs a full
1660 replacement of the resource representing the partial instance, satisfying the idempotency
1661 requirement for the PUT method demanded by RFC2616.

1662 If the $properties query parameter is not specified, the set of properties to be set is the set of all
1663 mutable properties of the target instance. If the $properties query parameter is specified, the set
1664 of properties to be set is the set of properties specified in the $properties query parameter.
1665 Properties specified in the Instance payload element that are not to be set as previously defined,
1666 shall be tolerated and ignored, even when they are not properties of the target instance.

1667 Mutable properties that are to be set as previously defined shall be set as specified for the property
1668 in the Instance payload element (including setting the property to Null), or if the property is not
1669 specified in the Instance payload element, to the class-defined default value of the property, or to
1670 Null if no such default value is defined.

1671 NOTE   This behavior for properties that are to be set but not specified in the Instance payload element is
1672 consistent with CIM-XML (DSP0200). In contrast, generic operations (DSP0223) requires that the property is set
1673 to Null in this case, even when a non-Null default value for the property is defined in the class.

1674 Requirements on mutability of properties can be defined in the model. Key properties are always
1675 unmutable.

1676 The "self", "namespace" and "classname" attributes in the request payload element are optional. If
1677 specified, they shall be consistent with the target resource identifier.

1678 On success, one of the following HTTP status codes shall be returned:

1679   • 204 "No Content": The entity body shall be empty.

1680 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
1681 the HTTP status codes in Table 15 or Table 7 shall be returned.

1682 **Table 15 – HTTP status codes for failing PUT (update an instance)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value, including the case: • the "self", "namespace" or "classname" attributes are not consistent with the target resource identifier
403	Forbidden	WIPG0249	Invalid input parameter value, including the cases: • a property specified in the `$properties` query parameter is unmutable • the new values for the properties violate requirements defined in the model
404	Not Found	WIPG0214	Class not found
501	Not Implemented	WIPG0228	Operation not supported by class implementation
404	Not Found	WIPG0213	Instance not found
403	Forbidden	WIPG0220	No such property, including the case: • a property specified in the `$properties` query parameter is not exposed by the creation class of the target instance

1683 **Example HTTP conversation for the full update of an instance (using JSON as defined in**
1684 **DSP0211):**

1685 Request (if type information is included in the request and accepted to be included in an error response):

```
1686  PUT /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11 HT
1687  TP/1.1
1688  Host: server.acme.com:5988
1689  Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
1690  Content-Length: XXX
1691  Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
1692  X-CIMRS-Version: 2.0.0
1693
1694  {
1695    "kind": "instance",
1696    "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
1697  ys11",
1698    "namespace": "root/cimv2",
1699    "classname": "ACME_VirtualSystem",
1700    "properties": {
1701      // InstanceID is not included because it is not updateable
1702      "ElementName": {
1703        "type": "string",
1704        "value": "Tom's system" },
1705      "Caption": {
```

```
1706          "type": "string",
1707          "value": "Tom's system (sys 11 on node 47)" },
1708        . . .      // all other updateable properties
1709      }
1710    }
```

1711   Response:

```
1712    HTTP/1.1 200 OK
1713    Date: Thu, 30 Oct 2014 15:03:00 GMT
1714    X-CIMRS-Version: 2.0.1
```

1715   NOTE   In this example, it is assumed that all provided properties are mutable. Because the set of properties to be
1716   changed has not been restricted using the `$properties` query parameter, the mutable properties not provided in
1717   the Instance payload element (for example, Description) are set to their class-defined default values or to Null. The
1718   value of the InstanceID key property remains unchanged, because key properties are never mutable.

1719   **Example HTTP conversation for the partial update of an instance (using JSON as defined in**
1720   **DSP0211):**

1721   Request (if type information is included in the request and accepted to be included in an error response):

```
1722    PUT /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11?$p
1723    roperties=ElementName,Caption HTTP/1.1
1724    Host: server.acme.com:5988
1725    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
1726    Content-Length: XXX
1727    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
1728    X-CIMRS-Version: 2.0.0
1729
1730    {
1731      "kind": "instance",
1732      "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
1733    ys11?$properties=ElementName,Caption",
1734      "namespace": "root/cimv2",
1735      "classname": "ACME_VirtualSystem",
1736      "properties": {
1737        "ElementName": {
1738          "type": "string",
1739          "value": "Tom's system" },
1740        "Caption": {
1741          "type": "string",
1742          "value": "Tom's system (sys 11 on node 47)" }
1743      }
1744    }
```

1745   Response:

```
1746    HTTP/1.1 200 OK
1747    Date: Thu, 30 Oct 2014 15:03:00 GMT
1748    X-CIMRS-Version: 2.0.1
```

1749   NOTE   In this example, it is assumed that all provided properties are mutable. Only the ElementName and Caption
1750   properties are set to their new values, because of the specified `$properties` query parameter.

1751   **7.5.7   DELETE (delete an instance)**

1752   **Purpose:**   Delete an instance

1753   **HTTP method:**   DELETE

1754   **Target resource:**   Instance (see 7.5.1)

1755   **Query parameters:**   None

1756   **Request headers:**   Host, Accept, X-CIMRS-Version

1757   **Request payload:**   None

1758   **Response headers (success):** Date, X-CIMRS-Version

1759   **Response payload (success):** None

1760   **Response headers (failure):**   Date, Content-Length, Content-Type, X-CIMRS-Version

1761   **Response payload (failure):**   ErrorResponse (see 7.3.5)

1762   **Requirement:**   Class-specific

1763   **Description:**

1764   The HTTP DELETE method on an instance resource deletes the instance resource, including the
1765   managed object represented by the instance resource.

1766   On success, one of the following HTTP status codes shall be returned:

1767   •   204 "No Content": The entity body shall be empty.

1768   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
1769   the HTTP status codes in Table 16 or Table 7 shall be returned.

1770   **Table 16 – HTTP status codes for failing DELETE (delete an instance)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value
404	Not Found	WIPG0214	Class not found
501	Not Implemented	WIPG0228	Operation not supported by class implementation
404	Not Found	WIPG0213	Instance not found

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
403	Forbidden	WIPG0246	Instance cannot be deleted due to referencing association
403	Forbidden	WIPG0247	Instance cannot be deleted due to multiplicity underflow

1771 **Example HTTP conversation (using JSON as defined in DSP0211):**

1772 Request (if type information is accepted to be included in an error response):

```
1773   DELETE /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11
1774    HTTP/1.1
1775   Host: server.acme.com:5988
1776   Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
1777   X-CIMRS-Version: 2.0.0
```

1778 Response:

```
1779   HTTP/1.1 204 No Content
1780   Date: Thu, 30 Oct 2014 15:03:00 GMT
1781   X-CIMRS-Version: 2.0.1
```

1782 ### 7.5.8 POST (invoke a method on an instance)

1783 **Purpose:**   Invoke a method on an instance

1784 **HTTP method:**   POST

1785 **Target resource:**   Instance (see 7.5.1)

1786 **Query parameters:**   None

1787 **Request headers:**   Host, Accept, Content-Length, Content-Type, X-CIMRS-Version

1788 **Request payload:**   MethodRequest (see 7.5.3)

1789 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

1790 **Response payload (success):** MethodResponse (see 7.5.4)

1791 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

1792 **Response payload (failure):** ErrorResponse (see 7.3.5)

1793 **Requirement:**   Class-specific

1794 **Description:**

1795 The HTTP POST method on an instance resource invokes the method specified in the
1796 MethodRequest payload element on that instance.

1797 The method may be static or non-static.

1798 On success, one of the following HTTP status codes shall be returned:

1799 • 200 "OK": The entity body shall contain a MethodResponse payload element (see Table
1800   13)

1801 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
1802 the HTTP status codes in Table 17 or Table 7 shall be returned.

1803 **Table 17 – HTTP status codes for failing POST (invoke a method on an instance)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0229	Method invocation not supported by WBEM server infrastructure
404	Not Found	WIPG0218	No such method
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value
404	Not Found	WIPG0214	Class not found
501	Not Implemented	WIPG0219	Method not supported by class implementation
404	Not Found	WIPG0213	Instance not found

1804 Note that the ErrorResponse payload element used on failure cannot represent method output
1805 parameters or a method return value.

1806   **Example HTTP conversation (using JSON as defined in DSP0211):**

1807   Request (if type information is included in the request and accepted to be included in the response):

```
1808    POST /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11 H
1809    TTP/1.1
1810    Host: server.acme.com:5988
1811    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
1812    Content-Length: XXX
1813    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
1814    X-CIMRS-Version: 2.0.0
1815
1816    {
1817      "kind": " methodrequest",
1818      "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
1819    ys11",
1820      "methodname": "RequestStateChange",
1821      "parameters": {
1822        "RequestedState": {
1823          "type": "uint16",
1824          "value": 2 },
1825        "TimeoutPeriod": {
1826          "type": "datetime",
1827          "value": None }
1828      }
1829    }
```

1830   Response (if type information is included):

```
1831    HTTP/1.1 200 OK
1832    Date: Thu, 30 Oct 2014 15:03:00 GMT
1833    Content-Length: XXX
1834    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
1835    X-CIMRS-Version: 2.0.1
1836
1837    {
1838      "kind": "methodresponse",
1839      "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
1840    ys11",
1841      "methodname": "RequestStateChange",
1842      "returnvalue": {
1843        "type": "uint32",
1844        "value": 0 },
1845      "parameters": {
1846        "Job": {
1847          "type": "reference",
1848          "classname": "ACME_Job",
1849          "value": None },
1850      }
1851    }
```

1852 **7.6 Instance collection resource**

1853 An instance collection resource represents a collection of instances of a particular class.

1854 **7.6.1 Resource identifier**

1855 Instance collection resources shall have a resource identifier whose path component (that is, the `path-`
1856 `absolute` ABNF rule in 6.1) matches ABNF rule `instance-coll-path-absolute`:

1857 `instance-coll-path-absolute = "/" nsname "/classes/" classname "/instances"`

1858 Where:

1859 • `nsname` is the namespace name, in its original lexical case, percent-encoded as defined in 6.3.
1860 The reserved character "/" in namespace names shall be considered data for purposes of
1861 percent-encoding (that is, it shall be percent-encoded); otherwise, namespace names do not
1862 contain reserved characters.

1863 • `classname` is the class name, in its original lexical case, percent-encoded as defined in 6.3.
1864 Note that CIM class names do not contain reserved characters (see 6.3 and DSP0004).

1865 Examples:

1866 `/root%2Fcimv2/classes/ACME_ComputerSystem/instances`

1867 **7.6.2 InstanceCollection payload element**

1868 An InstanceCollection payload element is the representation of an instance collection resource or
1869 instance collection page resource in the protocol.

1870 Unless otherwise constrained, an InstanceCollection payload element shall have the attributes defined in
1871 Table 18.

1872 **Table 18 – Attributes of an InstanceCollection payload element**

Attribute name	Payload data type	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value `"instancecollection"`
self	URI	Mandatory	resource identifier of the represented resource (instance collection or instance collection page).
next	URI	Conditional	resource identifier of the next instance collection page. Condition: Paged retrieval is used, and there are remaining pages in the paged retrieval stream Default if not specified: Paged retrieval is not used, or there are no more remaining pages in the paged retrieval stream.
instances	Instance [ ]	Mandatory	list of instances in the represented instance collection or instance collection page

1873 **7.6.3 POST (create an instance)**

1874 **Purpose:** Create an instance

1875 **HTTP method:** POST

1876 **Target resource:** Instance collection (see 7.6.1)

1877	**Query parameters:**	None
1878	**Request headers:**	Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
1879	**Request payload:**	Instance (see 7.5.2), without the "self" attribute
1880	**Response headers (success):**	Date, Location, X-CIMRS-Version
1881	**Response payload (success):**	None
1882	**Response headers (failure):**	Date, Content-Length, Content-Type, X-CIMRS-Version
1883	**Response payload (failure):**	ErrorResponse (see 7.3.5)
1884	**Requirement:**	Class-specific

1885 **Description:**

1886 The HTTP POST method on an instance collection resource creates an instance of the class of that
1887 collection, including any backing managed resource.

1888 On return, the Location header specifies the resource identifier of the newly created instance.

1889 The creation class of the new instance shall be the class of the collection resource that is targeted.

1890 The set of properties to be initialized in the new instance by the server is the set of all properties
1891 exposed by the creation class.

1892 Properties specified in the Instance payload element represent client-supplied initial values for the
1893 new instance.

1894 Properties specified in the Instance payload element that are not properties exposed by the creation
1895 class shall cause the server to fail the operation with HTTP status code 403 "Forbidden". Properties
1896 specified in the Instance payload element that are not client-initializable shall cause the server to fail
1897 the operation with HTTP status code 403 "Forbidden".

1898 Client-initializable properties shall be initialized as specified for the property in the Instance payload
1899 element (including initializing the property to Null), or if the property is not specified in the Instance
1900 payload element, to the class-defined default value of the property, or to Null if no such default value
1901 is defined.

1902 Any other properties of the instance shall be initialized as defined by the implementation, taking into
1903 account any requirements on the initial values defined in the model.

1904 The "self" attribute in the request payload element shall be omitted.

1905 The "namespace" and "classname" attributes in the request payload element are optional. If
1906 specified, they shall be consistent with the target resource identifier.

1907 On success, one of the following HTTP status codes shall be returned:

1908 • 201 "Created": The entity body shall be empty and the "Location" header field shall be set
1909 to the resource identifier of the newly created instance

1910 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
1911 the HTTP status codes in Table 19 or Table 7 shall be returned.

1912 **Table 19 – HTTP status codes for failing POST (create an instance)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value, including the following cases: • the "self" attribute is not omitted • the "namespace" or "classname" attributes are not consistent with the target resource identifier
403	Forbidden	WIPG0249	Invalid input parameter value, for the following cases: • a specified property is not client-initializable • the specified property values violate requirements defined in the model
404	Not Found	WIPG0249	Invalid input parameter value, for the following case: • a specified property is not exposed by the creation class of the new instance
404	Not Found	WIPG0214	Class not found
501	Not Implemented	WIPG0228	Operation not supported by class implementation
400	Bad Request	WIPG0216	Instance already exists

1913 **Example HTTP conversation (using JSON as defined in DSP0211):**

1914 Request (if type information is included in the request and accepted to be included in an error response):

```
1915    POST /root%2Fcimv2/classes/ACME_VirtualSystem/instances HTTP/1.1
1916    Host: server.acme.com:5988
1917    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
1918    Content-Length: XXX
1919    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
1920    X-CIMRS-Version: 2.0.0
1921
1922    {
1923      "kind": "instance",
1924      // self is omitted in creation
1925      "namespace": "root/cimv2",
1926      "classname": "ACME_VirtualSystem",
```

```
1927      "properties": {
1928        "ElementName": {
1929          "type": "string",
1930          "value": "Tom's system" },
1931        // Other initial property values
1932      }
1933    }
```

1934 Response:

```
1935    HTTP/1.1 201 Created
1936    Date: Thu, 30 Oct 2014 15:03:00 GMT
1937    Location: //server.acme.com:5988/root%2Fcimv2/classes/ACME_VirtualSystem/instances/
1938    InstanceID=node47%3Asys11
1939    X-CIMRS-Version: 2.0.1
```

1940   NOTE   The key property InstanceID is not provided in the request, because key property values are normally
1941   determined by the server. Other properties of the class (for example, Caption or Description) that are not
1942   provided by the client are initialized to their class-defined default values, or to Null.

1943   **7.6.4   GET (enumerate instances of a class)**

1944   **Purpose:**   Enumerate instances of a class

1945   **HTTP method:**   GET

1946   **Target resource:**   Instance collection (see 7.6.1)

1947   **Query parameters:**   `$properties`, `$filter`, `$filterlanguage`, `$continueonerror`,
1948   `$pagingtimeout`, `$max`

1949   **Request headers:**   Host, Accept, X-CIMRS-Version

1950   **Request payload:**   None

1951   **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

1952   **Response payload (success):** InstanceCollection (see 7.6.2), may be paged

1953   **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

1954   **Response payload (failure):** ErrorResponse (see 7.3.5)

1955   **Requirement:**   Class-specific

1956   **Description:**

1957   The HTTP GET method on an instance collection resource enumerates instances of the class of that
1958   collection (including instances of subclasses) and returns an instance collection (or subset thereof, if
1959   paged) with representations of these instances.

1960   The server may choose to use paging for the returned instance collection. For details on paged
1961   retrieval, see 7.3.7. If the server uses paging, the resource identifier for subsequent pages can be
1962   discovered from the "next" attribute of the current page. The next page can be retrieved using GET
1963   (see 7.9.2). A paged instance collection can be closed using DELETE (see 7.9.3).

1964   For details on the effects of the query parameters on the returned InstanceCollection payload
1965   element, see the descriptions of these query parameters in 6.6.

1966 On success, one of the following HTTP status codes shall be returned:

1967 • 200 "OK": The entity body shall contain an InstanceCollection payload element
1968 representing the returned instances (see 7.6.2). The collection may be empty.

1969 • 304 "Not Modified": The validators matched on a conditional request; the entity body shall
1970 be empty. This status code can only occur if the server supports conditional requests and
1971 the client has requested a conditional request. For details on conditional requests, see
1972 subclause 9.3 in RFC2616.

1973 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
1974 the HTTP status codes in Table 20 or Table 7 shall be returned.

1975 **Table 20 – HTTP status codes for failing GET (enumerate instances of a class)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0242	Invalid timeout
400	Bad Request	WIPG0249	Invalid input parameter value
400	Bad Request	WIPG0235	Continuation on error not supported
400	Bad Request	WIPG0237	Filter queries not supported by WBEM server infrastructure
400	Bad Request	WIPG0244	Filter queries not supported by class implementation
400	Bad Request	WIPG0221	Unknown query language
400	Bad Request	WIPG0222	Query language feature not supported
400	Bad Request	WIPG0223	Invalid query
404	Not Found	WIPG0214	Class not found
501	Not Implemented	WIPG0228	Operation not supported by class implementation

1976 **Example HTTP conversation (using JSON as defined in DSP0211):**

1977 Request (if type information is accepted to be included in the response):

```
1978   GET /root%2Fcimv2/classes/ACME_ComputerSystem/instances HTTP/1.1
1979   Host: server.acme.com:5988
1980   Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
1981   X-CIMRS-Version: 2.0.0
```

1982   Response (if type information is included, and server does not use paging):

```
1983   HTTP/1.1 200 OK
1984   Date: Thu, 30 Oct 2014 15:03:00 GMT
1985   Content-Length: XXX
1986   Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
1987   X-CIMRS-Version: 2.0.1
1988
1989   {
1990     "kind": "instancecollection",
1991     "self": "/root%2Fcimv2/classes/ACME_ComputerSystem/instances",
1992     "instances": [
1993       {
1994         "kind": "instance",
1995         "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47
1996   %3Asys11",
1997         "namespace": "root/cimv2",
1998         "classname": "ACME_ComputerSystem",
1999         "properties": {
2000           "InstanceID": {
2001             "type": "string",
2002             "value": "node47:sys11" },
2003           "ElementName": {
2004             "type": "string",
2005             "value": "Tom's system" },
2006           // Other property values of this instance
2007         }
2008       },
2009       // Other instances of this class
2010     ]
2011   }
```

2012   NOTE   This example assumes that ACME_VirtualSystem is a subclass of ACME_ComputerSystem.

2013   7.7   Instance associator collection resource

2014   An instance associator collection resource represents instances associated to a source instance.

2015   7.7.1   Resource identifier

2016 Instance associator collection resources shall have a resource identifier whose path component (that is,
2017 the `path-absolute` ABNF rule in 6.1) matches ABNF rule `instance-associator-coll-path-`
2018 `absolute`:

```
2019   instance-associator-coll-path-absolute = instance-path-absolute "/associators"
```

2020 Where:

2021   •   `instance-path-absolute` is the path component of the resource identifier of the source
2022   instance.

2023 ### 7.7.2   GET (retrieve associated instances)

2024 | **Purpose:** | Retrieve associated instances |

2025 | **HTTP method:** | GET |

2026 | **Target resource:** | Instance associator collection (see 7.7.1) |

2027 | **Query parameters:** | `$associationclass`, `$sourcerole`, `$associatedclass`,
2028 | | `$associatedrole`, `$properties`, `$filter`, `$filterlanguage`,
2029 | | `$continueonerror`, `$pagingtimeout`, `$max` |

2030 | **Request headers:** | Host, Accept, X-CIMRS-Version |

2031 | **Request payload:** | None |

2032 | **Response headers (success):** | Date, Content-Length, Content-Type, X-CIMRS-Version |

2033 | **Response payload (success):** | InstanceCollection (see 7.6.2), may be paged |

2034 | **Response headers (failure):** | Date, Content-Length, Content-Type, X-CIMRS-Version |

2035 | **Response payload (failure):** | ErrorResponse (see 7.3.5) |

2036 | **Requirement:** | Class-specific |

2037 **Description:**

2038 The HTTP GET method on an instance associator collection resource traverses associations starting
2039 on a source instance and returns an instance collection (or subset thereof, if paged) with
2040 representations of the instances associated with the source instance.

2041 The server may choose to use paging for the returned instance collection. For details on paged
2042 retrieval, see 7.3.7. If the server uses paging, the resource identifier for subsequent pages can be
2043 discovered from the "next" attribute of the current page. The next page can be retrieved using GET
2044 (see 7.9.2). A paged instance collection can be closed using DELETE (see 7.9.3).

2045 For details on the effects of the query parameters on the returned InstanceCollection payload
2046 element, see the descriptions of these query parameters in 6.6.

2047 On success, one of the following HTTP status codes shall be returned:

2048 - 200 "OK": The entity body shall contain an InstanceCollection payload element
2049   representing the returned instances (see 7.6.2). The collection may be empty.

2050 - 304 "Not Modified": The validators matched on a conditional request; the entity body shall
2051   be empty. This status code can only occur if the server supports conditional requests and
2052   the client has requested a conditional request. For details on conditional requests, see
2053   subclause 9.3 in RFC2616.

2054 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2055 the HTTP status codes in Table 21 or Table 7 shall be returned.

2056 **Table 21 – HTTP status codes for failing GET (retrieve associated instances)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0242	Invalid timeout
400	Bad Request	WIPG0249	Invalid input parameter value
400	Bad Request	WIPG0235	Continuation on error not supported
400	Bad Request	WIPG0237	Filter queries not supported by WBEM server infrastructure
400	Bad Request	WIPG0244	Filter queries not supported by class implementation
400	Bad Request	WIPG0221	Unknown query language
400	Bad Request	WIPG0222	Query language feature not supported
400	Bad Request	WIPG0223	Invalid query
404	Not Found	WIPG0214	Class not found
501	Not Implemented	WIPG0228	Operation not supported by class implementation

2057 **Example HTTP conversation (using JSON as defined in DSP0211):**

2058 Request (if type information is accepted to be included in the response):

```
2059   GET /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11/as
2060   sociators HTTP/1.1
2061   Host: server.acme.com:5988
2062   Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2063   X-CIMRS-Version: 2.0.0
```

2064 Response (if type information is included and server does not use paging):

```
2065   HTTP/1.1 200 OK
2066   Date: Thu, 30 Oct 2014 15:03:00 GMT
2067   Content-Length: XXX
2068   Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
2069   X-CIMRS-Version: 2.0.1
2070
2071   {
2072     "kind": "instancecollection",
2073     "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
2074   ys11/associators",
```

```
2075      "instances": [
2076        {
2077          "kind": "instance",
2078          "self": "/root%2Fcimv2/classes/ACME_NetworkInterface/instances/InstanceID=nod
2079    e47%3Asys11%3Aeth0",
2080          "namespace": "root/cimv2",
2081          "classname": "ACME_NetworkInterface",
2082          "properties": {
2083            "InstanceID": {
2084              "type": "string",
2085              "value": "eth0" },
2086            "IPAddress": {
2087              "type": "string",
2088              "value": "10.11.12.13" },
2089            . . .  // Other properties of this instance
2090          }
2091        },
2092        . . .  // Other associated instances
2093      ]
2094    }
```

7.8   Instance reference collection resource

A instance reference collection resource represents association instances referencing a source instance.

7.8.1   Resource identifier

Instance reference collection resources shall have a resource identifier whose path component (that is, the `path-absolute` ABNF rule in 6.1) matches ABNF rule `instance-reference-coll-path-absolute`:

```
instance-reference-coll-path-absolute = instance-path-absolute "/references"
```

Where:

- `instance-path-absolute` is the path component of the resource identifier of the source instance.

7.8.2   GET (retrieve referencing instances)

Purpose:	Retrieve referencing instances
HTTP method:	GET
Target resource:	Instance reference collection (see 7.8.1)
Query parameters:	$associationclass, $sourcerole, $properties, $filter, $filterlanguage, $continueonerror, $pagingtimeout, $max
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version

2114   **Response payload (success):** InstanceCollection (see 7.6.2), may be paged

2115   **Response headers (failure):**   Date, Content-Length, Content-Type, X-CIMRS-Version

2116   **Response payload (failure):**   ErrorResponse (see 7.3.5)

2117   **Requirement:**   Class-specific

2118   **Description:**

2119   The HTTP GET method on an instance reference collection resource traverses associations starting
2120   on a source instance and returns an instance collection (or subset thereof, if paged) with
2121   representations of the association instances that reference the source instance.

2122   The server may choose to use paging for the returned instance collection. For details on paged
2123   retrieval, see 7.3.7. If the server uses paging, the resource identifier for subsequent pages can be
2124   discovered from the "next" attribute of the current page. The next page can be retrieved using GET
2125   (see 7.9.2). A paged instance collection can be closed using DELETE (see 7.9.3).

2126   For details on the effects of the query parameters on the returned InstanceCollection payload
2127   element, see the descriptions of these query parameters in 6.6.

2128   On success, one of the following HTTP status codes shall be returned:

2129   •   200 "OK": The entity body shall contain an InstanceCollection payload element
2130   representing the returned instances (see 7.6.2). The collection may be empty.

2131   •   304 "Not Modified": The validators matched on a conditional request; the entity body shall
2132   be empty. This status code can only occur if the server supports conditional requests and
2133   the client has requested a conditional request. For details on conditional requests, see
2134   subclause 9.3 in RFC2616.

2135   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2136   the HTTP status codes in Table 22 or Table 7 shall be returned.

2137   **Table 22 – HTTP status codes for failing GET (retrieve referencing instances)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0242	Invalid timeout
400	Bad Request	WIPG0249	Invalid input parameter value
400	Bad Request	WIPG0235	Continuation on error not supported
400	Bad Request	WIPG0237	Filter queries not supported by WBEM server infrastructure

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
400	Bad Request	WIPG0244	Filter queries not supported by class implementation
400	Bad Request	WIPG0221	Unknown query language
400	Bad Request	WIPG0222	Query language feature not supported
400	Bad Request	WIPG0223	Invalid query
404	Not Found	WIPG0214	Class not found
501	Not Implemented	WIPG0228	Operation not supported by class implementation

2138 **Example HTTP conversation (using JSON as defined in DSP0211):**

2139 Request (if type information is accepted to be included in the response):

```
2140  GET /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11/re
2141  ferences HTTP/1.1
2142  Host: server.acme.com:5988
2143  Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2144  X-CIMRS-Version: 2.0.0
```

2145 Response (if type information is included and server does not use paging):

```
2146  HTTP/1.1 200 OK
2147  Date: Thu, 30 Oct 2014 15:03:00 GMT
2148  Content-Length: XXX
2149  Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
2150  X-CIMRS-Version: 2.0.1
2151
2152  {
2153    "kind": "instancecollection",
2154    "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
2155  ys11/references",
2156    "instances": [
2157      {
2158        "kind": "instance",
2159        "self": "/root%2Fcimv2/ACME_SystemNetworkDevice/System=. . .,Device=. . .",
2160        "namespace": "root/cimv2",
2161        "classname": "ACME_SystemNetworkDevice",
2162        "properties": {
2163          "System": {
2164            "type": "reference",
2165            "classname": "ACME_VirtualSystem",
2166            "value": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=n
2167  ode47%3Asys11" },
2168          "Device": {
2169            "type": "reference",
2170            "classname": "ACME_NetworkInterface",
2171            "value": "/root%2Fcimv2/classes/ACME_NetworkInterface/instances/InstanceI
```

```
2172       D=node47%3Asys11%3Aeth0" },
2173              . . .    // Other property values of this association instance
2174            }
2175          },
2176          . . .   // Other referencing association instances
2177        ]
2178      }
```

7.9   Instance collection page resource

2179

2180   An instance collection page resource represents a subsequent (second to last) page of a paged instance
2181   collection (see 7.6.1), paged instance associator collection (see 7.7.1), or paged instance reference
2182   collection (see 7.8.1).

7.9.1   Resource identifier

2183

2184   The resource identifier of instance collection page resources is server-implementation-specific. See 7.3.7
2185   for details.

7.9.2   GET (retrieve instance collection page)

2186

2187   **Purpose:**   Retrieve instance collection page

2188   **HTTP method:**   GET

2189   **Target resource:**   Instance collection page (see 7.9.1)

2190   **Query parameters:** $max

2191   **Request headers:**   Host, Accept, X-CIMRS-Version

2192   **Request payload:**   None

2193   **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

2194   **Response payload (success):** InstanceCollection (see 7.6.2)

2195   **Response headers (failure):**   Date, Content-Length, Content-Type, X-CIMRS-Version

2196   **Response payload (failure):**   ErrorResponse (see 7.3.5)

2197   **Requirement:**   Class-specific

2198   **Description:**

2199   The HTTP GET method on an instance collection page resource returns the next page of the paged
2200   instance collection.

2201   For details on paged retrieval, see 7.3.7.

2202   For details on the effects of the query parameters on the returned InstanceCollection payload
2203   element, see the descriptions of these query parameters in 6.6.

2204   On success, one of the following HTTP status codes shall be returned:

2205   •   200 "OK": The entity body shall contain an InstanceCollection payload element
2206   representing the returned instances (see 7.6.2). The collection may be empty.

2207
2208
On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 23 or Table 7 shall be returned.

2209
Table 23 – HTTP status codes for failing GET (retrieve instance collection page)

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value
501	Not Implemented	WIPG0228	Operation not supported by class implementation
404	Not Found	WIPG0241	Invalid enumeration context
404	Not Found	WIPG0238	Pull operation has been abandoned due to enumeration context closure

2210
Example HTTP conversation (using JSON as defined in DSP0211):

2211
Request (if type information is accepted to be included in the response):

2212
Note that the target resource identifier is server-implementation-specific.

```
2213   GET /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11/re
2214   ferences/page/123456 HTTP/1.1
2215   Host: server.acme.com:5988
2216   Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2217   X-CIMRS-Version: 2.0.0
```

2218
Response (if type information is included):

```
2219   HTTP/1.1 200 OK
2220   Date: Thu, 30 Oct 2014 15:03:00 GMT
2221   Content-Length: XXX
2222   Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
2223   X-CIMRS-Version: 2.0.1
2224
2225   {
2226     "kind": "instancecollection",
2227     "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
2228   ys11/references/page/123456",
2229     "instances": [
2230       {
2231         "kind": "instance",
2232         "self": "/root%2Fcimv2/ACME_SystemNetworkDevice/System=. . .,Device=. . .",
2233         "namespace": "root/cimv2",
```

```
2234          "classname": "ACME_SystemNetworkDevice",
2235          "properties": {
2236            "System": {
2237              "type": "reference",
2238              "classname": "ACME_VirtualSystem",
2239              "value": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=n
2240     ode47%3Asys11" },
2241            "Device": {
2242              "type": "reference",
2243              "classname": "ACME_NetworkInterface",
2244              "value": "/root%2Fcimv2/classes/ACME_NetworkInterface/instances/InstanceI
2245     D=node47%3Asys11%3Aeth0" },
2246            . . .  // Other property values of this instance
2247          }
2248        },
2249        . . .  // Other instances in this page
2250      ]
2251    }
```

2252 **7.9.3   DELETE (close paged instance collection)**

2253 **Purpose:**   Close paged instance collection

2254 **HTTP method:**   DELETE

2255 **Target resource:**   Instance collection page (see 7.9.1)

2256 **Query parameters:**   None

2257 **Request headers:**   Host, Accept, X-CIMRS-Version

2258 **Request payload:**   None

2259 **Response headers (success):** Date, X-CIMRS-Version

2260 **Response payload (success):** None

2261 **Response headers (failure):**   Date, Content-Length, Content-Type, X-CIMRS-Version

2262 **Response payload (failure):**   ErrorResponse (see 7.3.5)

2263 **Requirement:**   Class-specific

2264 **Description:**

2265 The HTTP DELETE method on an instance collection page resource closes the associated paged
2266 instance collection.

2267 For details on paged retrieval, see 7.3.7.

2268 For details on the effects of the query parameters on the returned InstanceCollection payload
2269 element, see the descriptions of these query parameters in 6.6.

2270 On success, one of the following HTTP status codes shall be returned:

2271 • 200 "OK": The entity body shall contain an InstanceCollection payload element
2272 representing the returned instances (see 7.6.2). The collection may be empty.

2273   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2274   the HTTP status codes in Table 24 or Table 7 shall be returned.

2275   **Table 24 – HTTP status codes for failing DELETE (close paged instance collection)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value
501	Not Implemented	WIPG0228	Operation not supported by class implementation
404	Not Found	WIPG0241	Invalid enumeration context
403	Forbidden	WIPG0239	Pull operation cannot be abandoned

2276   **Example HTTP conversation (using JSON as defined in DSP0211):**

2277   Request (if type information is accepted to be included in an error response):

2278   Note that the resource identifier of an instance collection page is sever-implementation-specific.

```
2279    DELETE /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11
2280    /references/page/123456 HTTP/1.1
2281    Host: server.acme.com:5988
2282    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2283    X-CIMRS-Version: 2.0.0
```

2284   Response:

```
2285    HTTP/1.1 200 OK
2286    Date: Thu, 30 Oct 2014 15:03:00 GMT
2287    X-CIMRS-Version: 2.0.1
```

2288   7.10 Class resource

2289   A class resource represents a definition of a class of managed objects supported by the managed
2290   environment.

2291   Because CIM-RS is model-neutral, the class definition defines a model for a type of resource, which in
2292   turn defines how that type of resource is exposed as instance resources, see 5.4.

2293 **7.10.1 Resource identifier**

2294 Class resources shall have a resource identifier whose path component (that is, the `path-absolute`
2295 ABNF rule in 6.1) matches ABNF rule `class-path-absolute`:

2296 `class-path-absolute = "/" nsname "/classes/" classname`

2297 Where:

2298 • `nsname` is the namespace name, in its original lexical case, percent-encoded as defined in 6.3.
2299 The reserved character "/" in namespace names shall be considered data for purposes of
2300 percent-encoding (that is, it shall be percent-encoded); otherwise, namespace names do not
2301 contain reserved characters.

2302 • `classname` is the class name, in its original lexical case, percent-encoded as defined in 6.3.
2303 Note that CIM class names do not contain reserved characters (see 6.3 and DSP0004).

2304 Examples:

2305 `/root%2Fcimv2/classes/ACME_ComputerSystem`

2306 **7.10.2 Class payload element**

2307 A class payload element is the representation of a class definition resource (and thus, of a managed
2308 object in the managed environment) in the protocol.

2309 Unless otherwise constrained, a Class payload element shall have the attributes defined in Table 25.

2310 **Table 25 – Attributes of a Class payload element**

Attribute name	Payload data type	Requirement	Description
Kind	String	Mandatory	format of the payload element; shall have the value "class"
Self	URI	Mandatory	resource identifier of the represented class
namespace	String	Mandatory	namespace name of the represented class
Name	String	Mandatory	class name of the represented class
superclassname	String	Conditional	name of the superclass of the represented class. Condition: The class has a superclass. Default if not specified: The class has no superclass.
qualifiers	QualifierValue [ ]	Conditional	unordered list of qualifier values (see 7.2.1). Condition: The payload element includes qualifier values.
properties	ElementDefinition [ ]	Conditional	unordered list of property definitions (see 7.2.1). Condition: The payload element includes property definitions.
methods	MethodDefinition [ ]	Conditional	unordered list of method definitions (see 7.2.1). Condition: The payload element includes method definitions.

2311 **7.10.3 GET (retrieve a class)**

2312 **Purpose:**   Retrieve a class

2313 **HTTP method:**   GET

2314 **Target resource:**   Class (see 7.10.1)

2315 **Query parameters:**   `$qualifiers`

2316 **Request headers:**   Host, Accept, X-CIMRS-Version

2317 **Request payload:**   None

2318 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

2319 **Response payload (success):** Class (see 7.10.2)

2320 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2321 **Response payload (failure):** ErrorResponse (see 7.3.5)

2322 **Requirement:**   Optional

2323 **Description:**

2324   The HTTP GET method on a class resource retrieves a representation of the specified class.

2325   For details on the effects of the query parameters on the returned Class payload element, see the
2326   descriptions of these query parameters in 6.6.

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

2328   • 200 "OK": The entity body shall contain a Class payload element representing the returned
2329   class (see 7.10.2).

2330   • 304 "Not Modified": The validators matched on a conditional request; the entity body shall
2331   be empty. This status code can only occur if the server supports conditional requests and
2332   the client has requested a conditional request. For details on conditional requests, see
2333   subclause 9.3 in RFC2616.

2334   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2335   the HTTP status codes in Table 26 or Table 7 shall be returned.

2336   **Table 26 – HTTP status codes for failing GET (retrieve a class)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value
404	Not Found	WIPG0214	Class not found

2337 **Example HTTP conversation (using JSON as defined in DSP0211):**

2338 Request (if type information is accepted to be included in the response):

```
2339    GET /root%2Fcimv2/classes/ACME_VirtualSystem HTTP/1.1
2340    Host: server.acme.com:5988
2341    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2342    X-CIMRS-Version: 2.0.0
```

2343 Response (if type information is included. Note that the inclusion of type information influences the
2344 representation of classes if a non-Null value is specified for the default value of properties that are
2345 embedded instances. For details, see DSP0211):

```
2346    HTTP/1.1 200 OK
2347    Date: Thu, 30 Oct 2014 15:03:00 GMT
2348    Content-Length: XXX
2349    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
2350    X-CIMRS-Version: 2.0.1
2351
2352    {
2353      "kind": "class",
2354      "self": "/root%2Fcimv2/classes/ACME_VirtualSystem",
2355      "namespace": "root/cimv2",
2356      "name": "ACME_VirtualSystem",
2357      "superclassname": "ACME_ComputerSystem",
2358      "qualifiers": {
2359        "Description": {
2360          "type": "string",
2361          "value": "A virtual system.\n . . ."
2362        },
2363        . . .  // Other qualifier values for this class
2364      },
2365      "properties": {
2366        "InstanceID": {
2367          "qualifiers" : { . . . },
2368          // array and arraysize are omitted
2369          "type": "string"
2370        },
2371        "ElementName": {
2372          "qualifiers" : { . . . },
2373          "default": "",
2374          // array and arraysize are omitted
2375          "type": "string"
```

```
2376          },
2377          . . .   // Other property definitions for this class
2378        },
2379        "methods": {
2380          "RequestStateChange": {
2381            "qualifiers" : { . . . },
2382            // array and arraysize are omitted
2383            "type": "uint32"
2384            "parameters": {
2385              "RequestedState": {
2386                "qualifiers" : { . . . },
2387                // array and arraysize are omitted
2388                "type": "uint16"
2389              },
2390              . . .   // Other parameters of this method
2391            }
2392          },
2393          . . .   // Other method definitions for this class
2394        }
2395      }
```

2396 7.10.4 PUT (update a class)

2397 **Purpose:**	Update a class
2398 **HTTP method:**	PUT
2399 **Target resource:**	Class (see 7.10.1)
2400 **Query parameters:**	None
2401 **Request headers:**	Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
2402 **Request payload:**	Class (see 7.10.2)
2403 **Response headers (success):**	Date, X-CIMRS-Version
2404 **Response payload (success):**	None
2405 **Response headers (failure):**	Date, Content-Length, Content-Type, X-CIMRS-Version
2406 **Response payload (failure):**	ErrorResponse (see 7.3.5)
2407 **Requirement:**	Optional

2408 **Description:**

2409 The HTTP PUT method on a class resource updates the entire resource with the specified class
2410 representation.

2411 The "self" and "namespace" attributes in the request payload element are optional. If specified, they
2412 shall be consistent with the target resource identifier.

2413 On success, one of the following HTTP status codes shall be returned:

2414   • 204 "No Content": The entity body shall be empty.

2415   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2416   the HTTP status codes in Table 27 or Table 7 shall be returned.

2417   **Table 27 – HTTP status codes for failing PUT (update a class)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value, including the case: • the "self" or "namespace" attributes are not consistent with the target resource identifier
404	Not Found	WIPG0214	Class not found
403	Forbidden	WIPG0226	Superclass not found
403	Forbidden	WIPG0231	Incompatible class modification

2418 **Example HTTP conversation (using JSON as defined in DSP0211):**

2419 Request (if type information is included in the request and accepted to be included in an error response.
2420 Note that the inclusion of type information influences the representation of classes if a non-Null value is
2421 specified for the default value of properties that are embedded instances. For details, see DSP0211):

```
2422    PUT /root%2Fcimv2/classes/ACME_VirtualSystem HTTP/1.1
2423    Host: server.acme.com:5988
2424    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2425    Content-Length: XXX
2426    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
2427    X-CIMRS-Version: 2.0.0
2428
2429    {
2430      "kind": "class",
2431      "self": "/root%2Fcimv2/classes/ACME_VirtualSystem",
2432      "namespace": "root/cimv2",
2433      "name": "ACME_VirtualSystem",
2434      "superclassname": "ACME_ComputerSystem",
2435      "qualifiers": { . . . },
2436      "properties": { . . . },
2437      "methods": { . . . }
2438    }
```

2439   Response:

```
2440    HTTP/1.1 200 OK
2441    Date: Thu, 30 Oct 2014 15:03:00 GMT
2442    X-CIMRS-Version: 2.0.1
```

2443   **7.10.5 DELETE (delete a class)**

2444   **Purpose:**   Delete a class

2445   **HTTP method:**   DELETE

2446   **Target resource:**   Class (see 7.10.1)

2447   **Query parameters:**   None

2448   **Request headers:**   Host, Accept, X-CIMRS-Version

2449   **Request payload:**   None

2450   **Response headers (success):** Date, X-CIMRS-Version

2451   **Response payload (success):** None

2452   **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2453   **Response payload (failure):** ErrorResponse (see 7.3.5)

2454   **Requirement:**   Optional

2455   **Description:**

2456   The HTTP DELETE method on an instance resource deletes the class resource.

2457   On success, one of the following HTTP status codes shall be returned:

2458   • 204 "No Content": The entity body shall be empty.

2459   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2460   the HTTP status codes in Table 28 or Table 7 shall be returned.

2461   **Table 28 – HTTP status codes for failing DELETE (delete a class)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value
404	Not Found	WIPG0214	Class not found

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
403	Forbidden	WIPG0224	Class has subclasses
403	Forbidden	WIPG0225	Class has instances
403	Forbidden	WIPG0230	Class has referencing association classes

2462 **Example HTTP conversation (using JSON as defined in DSP0211):**

2463 Request (if type information is accepted to be included in an error response):

```
2464    DELETE /root%2Fcimv2/classes/ACME_VirtualSystem HTTP/1.1
2465    Host: server.acme.com:5988
2466    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2467    X-CIMRS-Version: 2.0.0
```

2468 Response:

```
2469    HTTP/1.1 204 No Content
2470    Date: Thu, 30 Oct 2014 15:03:00 GMT
2471    X-CIMRS-Version: 2.0.1
```

2472 **7.10.6 POST (invoke a method on a class)**

2473 **Purpose:** Invoke a method on a class

2474 **HTTP method:** POST

2475 **Target resource:** Class (see 7.10.1)

2476 **Query parameters:** None

2477 **Request headers:** Host, Accept, Content-Length, Content-Type, X-CIMRS-Version

2478 **Request payload:** MethodRequest (see 7.5.3)

2479 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

2480 **Response payload (success):** MethodResponse (see 7.5.4)

2481 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2482 **Response payload (failure):** ErrorResponse (see 7.3.5)

2483 **Requirement:** Class-specific

2484 **Description:**

2485 The HTTP POST method on a class resource invokes the method specified in the MethodRequest
2486 payload element on that class.

2487 The method shall be static.

2488 On success, one of the following HTTP status codes shall be returned:

2489 • 200 "OK": The entity body shall contain a MethodResponse payload element (see 7.5.4).

2490 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2491 the HTTP status codes in Table 29 or Table 7 shall be returned.

2492 **Table 29 – HTTP status codes for failing POST (invoke a method on a class)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0229	Method invocation not supported by WBEM server infrastructure
404	Not Found	WIPG0218	No such method
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value, including the following case: • the method is not static
404	Not Found	WIPG0214	Class not found
501	Not Implemented	WIPG0219	Method not supported by class implementation
404	Not Found	WIPG0213	Instance not found

2493 Note that the ErrorResponse payload element used on failure cannot represent method output
2494 parameters or a method return value.

2495 **Example HTTP conversation for invocation of static method (using JSON as defined in DSP0211):**

2496 Request (if type information is included):

```
2497  POST /root%2Fcimv2/classes/ACME_VirtualSystem HTTP/1.1
2498  Host: server.acme.com:5988
2499  Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2500  Content-Length: XXX
2501  Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
2502  X-CIMRS-Version: 2.0.0
2503
2504  {
2505    "kind": "methodrequest",
2506    "self": "/root%2Fcimv2/classes/ACME_VirtualSystem",
2507    "method": "CreateVirtualSystem",
2508    "parameters": {
2509      "Template": {
2510        "type": "string",
2511        "value": "small" }
2512    }
2513  }
```

2514   Response (if type information is included):

```
2515    HTTP/1.1 200 OK
2516    Date: Thu, 30 Oct 2014 15:03:00 GMT
2517    Content-Length: XXX
2518    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
2519    X-CIMRS-Version: 2.0.1
2520
2521    {
2522      "kind": "methodresponse",
2523      "self": "/root%2Fcimv2/classes/ACME_VirtualSystem",
2524      "method": "CreateVirtualSystem",
2525      "returnvalue": {
2526        "type": "uint32",
2527        "value": 0 },
2528      "parameters": {
2529        "System": {
2530          "type": "reference",
2531          "classname": "ACME_VirtualSystem",
2532          "value": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node4
2533    7%3Asys12" }
2534      }
2535    }
```

2536   7.11 Class collection resource

2537   A class collection resource represents a list of class resources.

2538   7.11.1 Resource identifier

2539   Class collection resources shall have a resource identifier whose path component (that is, the `path-`
2540   `absolute` ABNF rule in 6.1) matches ABNF rule `class-coll-path-absolute`:

2541   `class-coll-path-absolute = "/" nsname "/classes"`

2542   Where:

2543   •   `nsname` is the namespace name, in its original lexical case, percent-encoded as defined in 6.3.
2544   The reserved character "/" in namespace names shall be considered data for purposes of
2545   percent-encoding (that is, it shall be percent-encoded); otherwise, namespace names do not
2546   contain reserved characters.

2547   Examples:

2548   `/root%2Fcimv2/classes`

2549   7.11.2 ClassCollection payload element

2550   A ClassCollection payload element is the representation of a class collection resource in the protocol.

2551   Unless otherwise constrained, a ClassCollection payload element shall have the attributes defined in
2552   Table 30.

2553 **Table 30 – Attributes of a ClassCollection payload element**

Attribute name	Payload data type	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value `"classcollection"`
self	URI	Mandatory	resource identifier of the represented class collection.
classes	Class [ ]	Conditional	unordered list of classes in the collection. Condition: The payload element includes classes.

2554 **7.11.3 POST (create a class)**

2555 **Purpose:** Create a class

2556 **HTTP method:** POST

2557 **Target resource:** Class collection (see 7.11.1)

2558 **Query parameters:** None

2559 **Request headers:** Host, Accept, Content-Length, Content-Type, X-CIMRS-Version

2560 **Request payload:** Class (see 7.10.2), without the "self" attribute

2561 **Response headers (success):** Date, Location, X-CIMRS-Version

2562 **Response payload (success):** None

2563 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2564 **Response payload (failure):** ErrorResponse (see 7.3.5)

2565 **Requirement:** Optional

2566 **Description:**

2567 The HTTP POST method on a class collection resource creates the specified class in the
2568 namespace of that class collection.

2569 On return, the Location header specifies the resource identifier of the newly created class.

2570 The qualifiers, properties and methods for the new class are defined in a class representation in the
2571 payload.

2572 The "self" attribute in the request payload element shall be omitted.

2573 The "namespace" attribute in the request payload element is optional. If specified, it shall be
2574 consistent with the target resource identifier.

2575 On success, one of the following HTTP status codes shall be returned:

2576 • 201 "Created": The entity body shall be empty and the "Location" header field shall be set
2577 to the resource identifier of the newly created class.

2578 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2579 the HTTP status codes in Table 31 or Table 7 shall be returned.

2580   **Table 31 – HTTP status codes for failing POST (create a class)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value, including the following cases: • the "self" attribute is not omitted • the "namespace" attribute is not consistent with the target resource identifier
400	Bad Request	WIPG0217	Class already exists
400	Bad Request	WIPG0226	Superclass not found

2581   **Example HTTP conversation (using JSON as defined in DSP0211):**

2582   Request (if type information is included in the request and accepted to be included in an error response.
2583   Note that the inclusion of type information influences the representation of classes if a non-Null value is
2584   specified for the default value of properties that are embedded instances. For details, see [DSP0211](#)):

```
2585      POST /root%2Fcimv2/classes HTTP/1.1
2586      Host: server.acme.com:5988
2587      Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2588      Content-Length: XXX
2589      Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
2590      X-CIMRS-Version: 2.0.0
2591
2592      {
2593        "kind": "class",
2594        // self is omitted in creation
2595        "namespace": "root/cimv2",
2596        "name": "ACME_VirtualSystem",
2597        "superclassname": "ACME_ComputerSystem",
2598        "qualifiers": { . . . },
2599        "properties": { . . . },
2600        "methods": { . . . }
2601      }
```

2602   Response:

```
2603      HTTP/1.1 201 Created
2604      Date: Thu, 30 Oct 2014 15:03:00 GMT
2605      Location: //server.acme.com:5988/root%2Fcimv2/classes/ACME_VirtualSystem
2606      X-CIMRS-Version: 2.0.1
```

2607   **7.11.4 GET (enumerate classes)**

2608   **Purpose:**   Enumerate classes

2609   **HTTP method:**   GET

2610   **Target resource:**   Class collection (see 7.11.1)

2611   **Query parameters:**   `$class`, `$subclasses`, `$qualifiers`

2612   **Request headers:**   Host, Accept, X-CIMRS-Version

2613   **Request payload:**   None

2614   **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

2615   **Response payload (success):** ClassCollection (see 7.11.2)

2616   **Response headers (failure):**   Date, Content-Length, Content-Type, X-CIMRS-Version

2617   **Response payload (failure):**   ErrorResponse (see 7.3.5)

2618   **Requirement:**   Optional

2619   **Description:**

2620   The HTTP GET method on a class collection resource enumerates top-level classes in a namespace
2621   or subclasses of a specified class.

2622   The set of classes included in the result depends on both the `$class` and `$subclasses` query
2623   parameters, as follows:

2624   •   An intermediate set of classes is determined, as follows: If query parameter `$class` (see 6.6.4)
2625   is specified, the direct subclasses of the specified class are in the intermediate set. Otherwise,
2626   the top-level classes in the namespace identified of the target resource identifier are in the
2627   intermediate set.

2628   •   The value of the `$subclasses` query parameter (6.6.13) governs whether the intermediate set
2629   of classes becomes the result set (if false), or (if true) is amended by the direct and indirect
2630   subclasses of each class in the intermediate set.

2631   Qualifier values shall be returned for each returned class resource if the `$qualifiers` parameter
2632   (6.6.11) evaluates to true.

2633   For details on the effects of the query parameters on the returned ClassCollection payload element,
2634   see the descriptions of these query parameters in 6.6.

2635   On success, one of the following HTTP status codes shall be returned:

2636   •   200 "OK": The entity body shall contain a ClassCollection payload element representing
2637   the returned classes (see 7.11.2). The collection may be empty.

2638   •   304 "Not Modified": The validators matched on a conditional request; the entity body shall
2639   be empty. This status code can only occur if the server supports conditional requests and
2640   the client has requested a conditional request. For details on conditional requests, see
2641   subclause 9.3 in RFC2616.

2642   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2643   the HTTP status codes in Table 32 or Table 7 shall be returned.

2644 **Table 32 – HTTP status codes for failing GET (enumerate classes)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
404	Not Found	WIPG0214	Class not found
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0242	Invalid timeout
400	Bad Request	WIPG0249	Invalid input parameter value

2645 **Example HTTP conversation for enumerating the direct subclasses of a class (using JSON as**
2646 **defined in DSP0211):**

2647 Request (if type information is accepted to be included in the response):

```
2648   GET /root%2Fcimv2/classes?$class=ACME_ComputerSystem HTTP/1.1
2649   Host: server.acme.com:5988
2650   Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2651   X-CIMRS-Version: 2.0.0
```

2652 Response (if type information is included. Note that the inclusion of type information influences the
2653 representation of classes if a non-Null value is specified for the default value of properties that are
2654 embedded instances. For details, see DSP0211):

```
2655   HTTP/1.1 200 OK
2656   Date: Thu, 30 Oct 2014 15:03:00 GMT
2657   Content-Length: XXX
2658   Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
2659   X-CIMRS-Version: 2.0.1
2660
2661   {
2662     "kind": "classcollection",
2663     "self": "/root%2Fcimv2/classes?$class=ACME_ComputerSystem",
2664     "classes": [
2665       {
2666         "kind": "class",
2667         "self": "/root%2Fcimv2/classes/ACME_VirtualSystem",
2668         "namespace": "root/cimv2",
2669         "name": "ACME_VirtualSystem",
2670         "superclassname": "ACME_ComputerSystem",
2671         "qualifiers": { . . . },
2672         "properties": { . . . },
2673         "methods": { . . . }
```

```
2674           },
2675           . . .   // Other direct subclasses of ACME_ComputerSystem
2676      ]
2677     }
```

7.12 Class associator collection resource

2678

2679 A class associator collection resource represents the classes associated to a source class.

7.12.1 Resource identifier

2680

2681 Class associator collection resources shall have a resource identifier whose path component (that is, the
2682 `path-absolute` ABNF rule in 6.1) matches ABNF rule `class-associator-coll-path-absolute`:

2683 `class-associator-coll-path-absolute = class-path-absolute "/associators"`

2684 Where:

2685 • `class-path-absolute` is the path component of the resource identifier of the source class.

7.12.2 GET (retrieve associated classes)

2686

2687 **Purpose:**   Retrieve associated classes

2688 **HTTP method:**   GET

2689 **Target resource:**   Class associator collection (see 7.12.1)

2690 **Query parameters:**   `$associationclass`, `$sourcerole`, `$associatedclass`,
2691   `$associatedrole`, `$qualifiers`

2692 **Request headers:**   Host, Accept, X-CIMRS-Version

2693 **Request payload:**   None

2694 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

2695 **Response payload (success):** ClassCollection (see 7.11.2), may be paged

2696 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2697 **Response payload (failure):** ErrorResponse (see 7.3.5)

2698 **Requirement:**   Optional

2699 **Description:**

2700   The HTTP GET method on a class associator collection resource analyzes the class structure
2701   starting on a source class and returns a class collection with representations of the classes
2702   associated with the source class.

2703   For details on the effects of the query parameters on the returned ClassCollection payload element,
2704   see the descriptions of these query parameters in 6.6.

2705   On success, one of the following HTTP status codes shall be returned:

2706   • 200 "OK": The entity body shall contain a ClassCollection payload element representing
2707   the returned classes (see 7.11.2). The collection may be empty.

2708   • 304 "Not Modified": The validators matched on a conditional request; the entity body shall
2709   be empty. This status code can only occur if the server supports conditional requests and
2710   the client has requested a conditional request. For details on conditional requests, see
2711   subclause 9.3 in RFC2616.

2712   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2713   the HTTP status codes in Table 33 or Table 7 shall be returned.

2714   **Table 33 – HTTP status codes for failing GET (retrieve associated classes)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0242	Invalid timeout
400	Bad Request	WIPG0249	Invalid input parameter value

2715   **Example HTTP conversation (using JSON as defined in DSP0211):**

2716   Request (if type information is accepted to be included in the response):

```
2717    GET /root%2Fcimv2/classes/ACME_ComputerSystem/associators HTTP/1.1
2718    Host: server.acme.com:5988
2719    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2720    X-CIMRS-Version: 2.0.0
```

2721   Response (if type information is included. Note that the inclusion of type information influences the
2722   representation of classes if a non-Null value is specified for the default value of properties that are
2723   embedded instances. For details, see DSP0211):

```
2724    HTTP/1.1 200 OK
2725    Date: Thu, 30 Oct 2014 15:03:00 GMT
2726    Content-Length: XXX
2727    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
2728    X-CIMRS-Version: 2.0.1
2729
2730    {
2731      "kind": "classcollection",
2732      "self": "/root%2Fcimv2/classes/ACME_ComputerSystem/associators",
2733      "classes": [
```

```
2734          {
2735            "kind": "class",
2736            "self": "/root%2Fcimv2/classes/ACME_NetworkInterface",
2737            "namespace": "root/cimv2",
2738            "name": "ACME_NetworkInterface",
2739            "superclassname": "ACME_Device",
2740            "qualifiers": { . . . },
2741            "properties": { . . . },
2742            "methods": { . . . }
2743          },
2744          . . .  // Other associated classes
2745        ]
2746      }
```

7.13 Class reference collection resource

2748 A class reference collection resource represents the association classes referencing a source class.

7.13.1 Resource identifier

2750 Class reference collection resources shall have a resource identifier whose path component (that is, the
2751 `path-absolute` ABNF rule in 6.1) matches ABNF rule `class-reference-coll-path-absolute`:

```
2752 class-reference-coll-path-absolute = class-path-absolute "/references"
```

2753 Where:

2754 • `class-path-absolute` is the path component of the resource identifier of the source class.

7.13.2 GET (retrieve referencing classes)

2756 **Purpose:**	Retrieve referencing classes
2757 **HTTP method:**	GET
2758 **Target resource:**	Class reference collection (see 7.13.1)
2759 **Query parameters:**	`$associationclass`, `$sourcerole`, `$qualifiers`
2760 **Request headers:**	Host, Accept, X-CIMRS-Version
2761 **Request payload:**	None
2762 **Response headers (success):**	Date, Content-Length, Content-Type, X-CIMRS-Version
2763 **Response payload (success):**	ClassCollection (see 7.11.2), may be paged
2764 **Response headers (failure):**	Date, Content-Length, Content-Type, X-CIMRS-Version
2765 **Response payload (failure):**	ErrorResponse (see 7.3.5)
2766 **Requirement:**	Optional

2767   **Description:**

2768   The HTTP GET method on a class reference collection resource analyzes the class structure starting
2769   on a source class and returns a class collection with representations of the association classes
2770   referencing the source class.

2771   For details on the effects of the query parameters on the returned ClassCollection payload element,
2772   see the descriptions of these query parameters in 6.6.

2773   On success, one of the following HTTP status codes shall be returned:

2774   • 200 "OK": The entity body shall contain a ClassCollection payload element representing
2775   the returned classes (see 7.11.2). The collection may be empty.

2776   • 304 "Not Modified": The validators matched on a conditional request; the entity body shall
2777   be empty. This status code can only occur if the server supports conditional requests and
2778   the client has requested a conditional request. For details on conditional requests, see
2779   subclause 9.3 in RFC2616.

2780   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2781   the HTTP status codes in Table 34 or Table 7 shall be returned.

2782   **Table 34 – HTTP status codes for failing GET (retrieve referencing classes)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0242	Invalid timeout
400	Bad Request	WIPG0249	Invalid input parameter value

2783   **Example HTTP conversation (using JSON as defined in DSP0211):**

2784   Request (if type information is accepted to be included in the response):

```
2785    GET /root%2Fcimv2/classes/ACME_ComputerSystem/references HTTP/1.1
2786    Host: server.acme.com:5988
2787    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2788    X-CIMRS-Version: 2.0.0
```

2789   Response (if type information is included. Note that the inclusion of type information influences the
2790   representation of classes if a non-Null value is specified for the default value of properties that are
2791   embedded instances. For details, see DSP0211):

```
2792    HTTP/1.1 200 OK
2793    Date: Thu, 30 Oct 2014 15:03:00 GMT
2794    Content-Length: XXX
```

```
2795    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
2796    X-CIMRS-Version: 2.0.1
2797
2798    {
2799      "kind": "classcollection",
2800      "self": "/root%2Fcimv2/classes/ACME_ComputerSystem/references",
2801      "classes": [
2802        {
2803          "kind": "class",
2804          "self": "/root%2Fcimv2/classes/ACME_SystemDevice",
2805          "namespace": "root/cimv2",
2806          "name": "ACME_SystemDevice",
2807          // no superclass
2808          "qualifiers": { . . . },
2809          "properties": { . . . },
2810          // no methods
2811        },
2812        . . .  // Other referencing classes
2813      ]
2814    }
```

2815 7.14 Qualifier type resource

2816 A qualifier type resource represents a CIM qualifier type (that is, the declaration of a qualifier).

2817 7.14.1 Resource identifier

2818 Qualifier type resources shall have a resource identifier whose path component (that is, the `path-`
2819 `absolute` ABNF rule in 6.1) matches ABNF rule `qualifiertype-path-absolute`:

```
2820    qualifiertype-path-absolute = "/" nsname "/qualifiertypes/" qualifiername
```

2821 Where:

2822 • `nsname` is the namespace name, in its original lexical case, percent-encoded as defined in 6.3.
2823 The reserved character "/" in namespace names shall be considered data for purposes of
2824 percent-encoding (that is, it shall be percent-encoded); otherwise, namespace names do not
2825 contain reserved characters.

2826 • `qualifiername` is the qualifier name, percent-encoded as defined in 6.3. Note that CIM
2827 qualifier names do not contain reserved characters (see 6.3 and [DSP0004](#)).

2828 Examples:

```
2829    /root%2Fcimv2/qualifiertypes/Abstract
```

2830 7.14.2 QualifierType payload element

2831 A QualifierType payload element is the representation of a qualifier type in the protocol.

2832 Unless otherwise constrained, a QualifierType payload element shall have the attributes defined in Table
2833 35.

2834 **Table 35 – Attributes of a QualifierType payload element**

Attribute name	Payload data type	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value `"qualifiertype"`
self	URI	Mandatory	resource identifier of the represented qualifier type
namespace	String	Mandatory	namespace name of the represented qualifier type
name	String	Mandatory	name of the qualifier type
array	Boolean	Conditional	specifies whether the qualifier type is an array. Condition: The qualifier type is an array. Default if not specified: False.
type	String	Mandatory	CIM data type of the qualifier type
defaultvalue	Value	Conditional	default value for the qualifier. Condition: The default value is non-Null. Default if not specified: Null.
scopes	String [ ]	Mandatory	unordered list of scopes of the qualifier type. The set of scope values shall be the set defined in the description of the "Scope" attribute of the "Qualifier Type" metaelement in DSP0004. Scope values shall be compared case sensitively in CIM-RS.
propagation	Boolean	Mandatory	indicates whether qualifier values are propagated to subclasses. See the description of the "InheritancePropagation" attribute of the "Flavor" metaelement in DSP0004.
override	Boolean	Conditional	indicates whether qualifier values can be overridden in subclasses. See the description of the "OverridePermission" attribute of the "Flavor" metaelement in DSP0004. Condition: propagation is True. Default if not specified: Not applicable.
translatable	Boolean	Conditional	indicates whether qualifier values are translatable. See the description of the "Translatable" attribute of the "Flavor" metaelement in DSP0004. Condition: Qualifier values are translatable. Default if not specified: False.

2835 **7.14.3 GET (retrieve a qualifier type)**

2836 **Purpose:** Retrieve a qualifier type

2837 **HTTP method:** GET

2838 **Target resource:** Qualifier type (see 7.14.1)

2839 **Query parameters:** None

2840 **Request headers:** Host, Accept, X-CIMRS-Version

2841 **Request payload:** None

2842 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

2843   **Response payload (success):** QualifierType (see 7.14.2)

2844   **Response headers (failure):**   Date, Content-Length, Content-Type, X-CIMRS-Version

2845   **Response payload (failure):**   ErrorResponse (see 7.3.5)

2846   **Requirement:**   Optional

2847   **Description:**

2848   The HTTP GET method on a qualifier type resource retrieves a representation of the specified
2849   qualifier type.

2850   For details on the effects of the query parameters on the returned QualifierType payload element,
2851   see the descriptions of these query parameters in 6.6.

2852   On success, one of the following HTTP status codes shall be returned:

2853   •   200 "OK": The entity body shall contain a QualifierType payload element representing the
2854   returned qualifier type (see 7.14.2).

2855   •   304 "Not Modified": The validators matched on a conditional request; the entity body shall
2856   be empty. This status code can only occur if the server supports conditional requests and
2857   the client has requested a conditional request. For details on conditional requests, see
2858   subclause 9.3 in RFC2616.

2859   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2860   the HTTP status codes in Table 36 or Table 7 shall be returned.

2861   **Table 36 – HTTP status codes for failing GET (retrieve a qualifier type)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value
404	Not Found	WIPG0215	Qualifier type not found

2862   **Example HTTP conversation (using JSON as defined in DSP0211):**

2863   Request (if type information is accepted to be included in the response):

```
2864   GET /root%2Fcimv2/qualifiertypes/Abstract HTTP/1.1
2865   Host: server.acme.com:5988
2866   Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2867   X-CIMRS-Version: 2.0.0
```

2868 Response (if type information is included. Note that the inclusion of type information does not influence
2869 the representation of qualifier types. For details, see [DSP0211](#)):

```
2870    HTTP/1.1 200 OK
2871    Date: Thu, 30 Oct 2014 15:03:00 GMT
2872    Content-Length: XXX
2873    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
2874    X-CIMRS-Version: 2.0.1
2875
2876    {
2877      "kind": "qualifiertype",
2878      "self": "/root%2Fcimv2/qualifiertypes/Abstract",
2879      "namespace": "root/cimv2",
2880      "name": "Abstract",
2881      "type": "boolean",
2882      "scopes": ["class", "association", "indication"],
2883      "propagation": false,
2884      // override is omitted
2885      // translatable is omitted
2886    }
```

2887 **7.14.4 PUT (update a qualifier type)**

2888 **Purpose:**   Update a qualifier type

2889 **HTTP method:**   PUT

2890 **Target resource:**   Qualifier type (see 7.14.1)

2891 **Query parameters:**   None

2892 **Request headers:**   Host, Accept, Content-Length, Content-Type, X-CIMRS-Version

2893 **Request payload:**   QualifierType (see 7.14.2)

2894 **Response headers (success):** Date, X-CIMRS-Version

2895 **Response payload (success):** None

2896 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2897 **Response payload (failure):** ErrorResponse (see 7.3.5)

2898 **Requirement:**   Optional

2899 **Description:**

2900 The HTTP PUT method on a qualifier type resource updates the entire resource with the specified
2901 qualifier type representation.

2902 The "self" and "namespace" attributes in the request payload element are optional. If specified, they
2903 shall be consistent with the target resource identifier.

2904 On success, one of the following HTTP status codes shall be returned:

2905   • 204 "No Content": The entity body shall be empty.

2906   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2907   the HTTP status codes in Table 37 or Table 7 shall be returned.

2908   **Table 37 – HTTP status codes for failing PUT (update a qualifier type)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value, including the case: • the "self" or "namespace" attributes are not consistent with the target resource identifier
404	Not Found	WIPG0215	Qualifier type not found
403	Forbidden	WIPG0234	Incompatible modification of qualifier type
403	Forbidden	WIPG0245	Qualifier type inconsistent with DSP0004

2909   **Example HTTP conversation (using JSON as defined in DSP0211):**

2910   Request (if type information is included in the request and accepted to be included in an error response.
2911   Note that the inclusion of type information does not influence the representation of qualifier types. For
2912   details, see DSP0211):

```
2913    PUT /root%2Fcimv2/Abstract HTTP/1.1
2914    Host: server.acme.com:5988
2915    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2916    Content-Length: XXX
2917    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
2918    X-CIMRS-Version: 2.0.0
2919
2920    {
2921      "kind": "qualifiertype",
2922      "self": "/root%2Fcimv2/qualifiertypes/Abstract",
2923      "namespace": "root/cimv2",
2924      "name": "Abstract",
2925      "type": "boolean",
2926      "scopes": ["class", "association", "indication"],
2927      "propagation": false,
2928      // override is omitted
2929      // translatable is omitted
2930    }
```

2931   Response:

```
2932    HTTP/1.1 200 OK
2933    Date: Thu, 30 Oct 2014 15:03:00 GMT
2934    X-CIMRS-Version: 2.0.1
```

2935   **7.14.5 DELETE (delete a qualifier type)**

2936   **Purpose:**   Delete a qualifier type

2937   **HTTP method:**   DELETE

2938   **Target resource:**   Qualifier type (see 7.14.1)

2939   **Query parameters:**   None

2940   **Request headers:**   Host, Accept, X-CIMRS-Version

2941   **Request payload:**   None

2942   **Response headers (success):** Date, X-CIMRS-Version

2943   **Response payload (success):** None

2944   **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2945   **Response payload (failure):** ErrorResponse (see 7.3.5)

2946   **Requirement:**   Optional

2947   **Description:**

2948   The HTTP DELETE method on a qualifier type resource deletes the qualifier type in its namespace.

2949   On success, one of the following HTTP status codes shall be returned:

2950   •   204 "No Content": The entity body shall be empty.

2951   On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
2952   the HTTP status codes in Table 38 or Table 7 shall be returned.

2953   **Table 38 – HTTP status codes for failing DELETE (delete a qualifier type)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0215	Qualifier type not found
403	Forbidden	WIPG0233	Qualifier type is used

2954   **Example HTTP conversation (using JSON as defined in DSP0211):**

2955   Request (if type information is accepted to be included in an error response):

```
2956    DELETE /root%2Fcimv2/qualifiertypes/Abstract HTTP/1.1
2957    Host: server.acme.com:5988
2958    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
2959    X-CIMRS-Version: 2.0.0
```

2960   Response:

```
2961    HTTP/1.1 204 No Content
2962    Date: Thu, 30 Oct 2014 15:03:00 GMT
2963    X-CIMRS-Version: 2.0.1
```

2964   7.15 Qualifier type collection resource

2965   A qualifier type collection resource represents a list of qualifier types.

2966   7.15.1 Resource identifier

2967   Qualifier type collection resources shall have a resource identifier whose path component (that is, the
2968   `path-absolute` ABNF rule in 6.1) matches ABNF rule `qualifiertype-coll-path-absolute`:

```
2969    qualifiertype-coll-path-absolute = "/" nsname "/qualifiertypes"
```

2970   Where:

2971   •   `nsname` is the namespace name, in its original lexical case, percent-encoded as defined in 6.3.
2972   The reserved character "/" in namespace names shall be considered data for purposes of
2973   percent-encoding (that is, it shall be percent-encoded); otherwise, namespace names do not
2974   contain reserved characters.

2975   Examples:

```
2976    /root%2Fcimv2/qualifiertypes
```

2977   7.15.2 QualifierTypeCollection payload element

2978   A QualifierTypeCollection payload element is the representation of a qualifier type collection resource in
2979   the protocol.

2980   Unless otherwise constrained, a QualifierTypeCollection payload element shall have the attributes
2981   defined in Table 39.

2982 **Table 39 – Attributes of a QualifierTypeCollection payload element**

Attribute name	Payload data type	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value `"classcollection"`
self	URI	Mandatory	resource identifier of the represented qualifier type collection
qualifiertypes	QualifierType [ ]	Conditional	unordered list of qualifier types in the collection. Condition: The payload element includes qualifier types.

2983 ### 7.15.3 POST (create a qualifier type)

2984 **Purpose:** Create a qualifier type

2985 **HTTP method:** POST

2986 **Target resource:** Qualifier type collection (see 7.15.1)

2987 **Query parameters:** None

2988 **Request headers:** Host, Accept, Content-Length, Content-Type, X-CIMRS-Version

2989 **Request payload:** QualifierType (see 7.14.2), without the "self" attribute

2990 **Response headers (success):** Date, Location, X-CIMRS-Version

2991 **Response payload (success):** None

2992 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

2993 **Response payload (failure):** ErrorResponse (see 7.3.5)

2994 **Requirement:** Optional

2995 **Description:**

2996 The HTTP POST method on a qualifier type collection resource creates the specified qualifier type in
2997 the namespace of that collection.

2998 On return, the Location header specifies the resource identifier of the newly created qualifier type.

2999 The attributes for the new qualifier type are defined in a qualifier type representation in the payload.

3000 The "self" attribute in the request payload element shall be omitted.

3001 The "namespace" attribute in the request payload element is optional. If specified, it shall be
3002 consistent with the target resource identifier.

3003 On success, one of the following HTTP status codes shall be returned:

3004 • 201 "Created": The entity body shall be empty and the "Location" header field shall be set
3005 to the resource identifier of the newly created qualifier type.

3006 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
3007 the HTTP status codes in Table 40 or Table 7 shall be returned.

3008   **Table 40 – HTTP status codes for failing POST (create a qualifier type)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value, including the following cases: • the "self" attribute is not omitted • the "namespace" attribute is not consistent with the target resource identifier
403	Forbidden	WIPG0245	Qualifier type inconsistent with DSP0004

3009 **Example HTTP conversation (using JSON as defined in DSP0211):**

3010 Request (if type information is included in the request and accepted to be included in an error response.
3011 Note that the inclusion of type information does not influence the representation of qualifier types. For
3012 details, see DSP0211):

```
3013    POST /root%2Fcimv2/qualifiertypes HTTP/1.1
3014    Host: server.acme.com:5988
3015    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
3016    Content-Length: XXX
3017    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
3018    X-CIMRS-Version: 2.0.0
3019
3020    {
3021      "kind": "qualifiertype",
3022      // self is omitted in creation
3023      "namespace": "root/cimv2",
3024      "name": "Abstract",
3025      "type": "boolean",
3026      "scopes": ["class", "association", "indication"],
3027      "propagation": false,
3028      // override is omitted
3029      // translatable is omitted
3030    }
```

3031 Response:

```
3032    HTTP/1.1 201 Created
3033    Date: Thu, 30 Oct 2014 15:03:00 GMT
3034    Location: //server.acme.com:5988/root%2Fcimv2/qualifiertypes/Abstract
3035    X-CIMRS-Version: 2.0.1
```

3036 **7.15.4 GET (enumerate qualifier types)**

3037 **Purpose:** Enumerate qualifier types

3038 **HTTP method:** GET

3039 **Target resource:** Qualifier type collection (see 7.15.1)

3040 **Query parameters:** None

3041 **Request headers:** Host, Accept, X-CIMRS-Version

3042 **Request payload:** None

3043 **Response headers (success):** Date, Content-Length, Content-Type, X-CIMRS-Version

3044 **Response payload (success):** QualifierTypeCollection (see 7.15.2)

3045 **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

3046 **Response payload (failure):** ErrorResponse (see 7.3.5)

3047 **Requirement:** Optional

3048 **Description:**

3049 The HTTP GET method on a qualifier type collection resource enumerates the qualifier types in the
3050 namespace of that collection.

3051 For details on the effects of the query parameters on the returned QualifierTypeCollection payload
3052 element, see the descriptions of these query parameters in 6.6.

3053 On success, one of the following HTTP status codes shall be returned:

3054 • 200 "OK": The entity body shall contain a QualifierTypeCollection payload element
3055 representing the returned qualifier type (see 7.15.2). The collection may be empty.

3056 • 304 "Not Modified": The validators matched on a conditional request; the entity body shall
3057 be empty. This status code can only occur if the server supports conditional requests and
3058 the client has requested a conditional request. For details on conditional requests, see
3059 subclause 9.3 in RFC2616.

3060 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
3061 the HTTP status codes in Table 41 or Table 7 shall be returned.

3062 **Table 41 – HTTP status codes for failing GET (enumerate qualifier types)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
404	Not Found	WIPG0204	Namespace not found
501	Not Implemented	WIPG0203	Operation not supported by WBEM server infrastructure
404	Not Found	WIPG0214	Class not found
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0242	Invalid timeout
400	Bad Request	WIPG0249	Invalid input parameter value

3063 **Example HTTP conversation (using JSON as defined in DSP0211):**

3064 Request (if type information is accepted to be included in the response):

```
3065    GET /root%2Fcimv2/qualifiertypes HTTP/1.1
3066    Host: server.acme.com:5988
3067    Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
3068    X-CIMRS-Version: 2.0.1
```

3069 Response (if type information is included. Note that the inclusion of type information does not influence
3070 the representation of qualifier types. For details, see DSP0211):

```
3071    HTTP/1.1 200 OK
3072    Date: Thu, 30 Oct 2014 15:03:00 GMT
3073    Content-Length: XXX
3074    Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
3075    X-CIMRS-Version: 2.0.0
3076
3077    {
3078      "kind": "qualifiertypecollection",
3079      "self": "/root%2Fcimv2/qualifiertypes",
3080      "qualifiertypes": [
3081        {
3082          "kind": "qualifiertype",
3083          "self": "/root%2Fcimv2/qualifiertypes/Abstract",
3084          "namespace": "root/cimv2",
3085          "name": "Abstract",
3086          "type": "boolean",
3087          "scopes": ["class", "association", "indication"],
3088          "propagation": false,
3089          // override is omitted
3090          // translatable is omitted
3091        },
3092        . . .  // Other qualifier types in this namespace
3093      ]
3094    }
```

3095 ## 7.16 Listener indication delivery resource

3096 A listener indication delivery resource in a listener represents the ability to deliver an indication to the
3097 listener.

3098   **7.16.1 Resource identifier**

3099   Listener indication delivery resources shall have a resource identifier whose path component (that is, the
3100   `path-absolute` ABNF rule in 6.1) matches ABNF rule `listener-indications-path-absolute`:

3101   `listener-indications-path-absolute = "/destinations/" destname "/indications"`

3102   Where:

3103   • `destname` is the name of the listener destination, percent-encoded as defined in 6.3

3104   Examples:

3105   `/destinations/srv8%3Adest1/indications`

3106   **7.16.2 IndicationDeliveryRequest payload element**

3107   An IndicationDeliveryRequest payload element is the representation of a request to deliver an indication
3108   to a listener in the protocol.

3109   Unless otherwise constrained, an IndicationDeliveryRequest payload element shall have the attributes
3110   defined in Table 42.

3111   **Table 42 – Attributes of an IndicationDeliveryRequest payload element**

Attribute name	Payload data type	Requirement	Description
kind	String	Mandatory	format of the payload element; shall have the value `"indicationdeliveryrequest"`
self	URI	Mandatory	resource identifier of the listener indication delivery resource
indication	Instance	Mandatory	an embedded instance of a class that is an indication, specifying the indication to be delivered, with attributes "self" and "namespace" omitted

3112   **7.16.3 POST (deliver an indication)**

3113   **Purpose:**   Deliver an indication

3114   **HTTP method:**   POST

3115   **Target resource:**   Listener indication delivery (see 7.16.1)

3116   **Query parameters:**   None

3117   **Request headers:**   Host, Accept, Content-Length, Content-Type, X-CIMRS-Version

3118   **Request payload:**   IndicationDeliveryRequest (see 7.16.2)

3119   **Response headers (success):** Date, X-CIMRS-Version

3120   **Response payload (success):** None

3121   **Response headers (failure):** Date, Content-Length, Content-Type, X-CIMRS-Version

3122   **Response payload (failure):** ErrorResponse (see 7.3.5)

3123   **Requirement:**   Mandatory

3124 **Description:**

3125 The HTTP POST method on a listener indication delivery resource delivers an indication to the
3126 listener specified in that resource.

3127 Note that for this operation, the server decides which payload representation to use, and in case of
3128 using DSP0211, whether to include type information. In any case, the Content-Type header needs to
3129 be consistent with those decisions.

3130 For implementations supporting the event model defined in the CIM Schema published by DMTF, the
3131 target resource identifier for this operation is the value of the Destination property of
3132 CIM_ListenerDestination instances that indicate the CIM-RS protocol in their Protocol property. For
3133 details, see the *DMTF Indications Profile* (DSP1054).

3134 On success, one of the following HTTP status codes shall be returned:

3135 • 200 "OK": The entity body shall be empty.

3136 On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of
3137 the HTTP status codes in Table 43 or Table 7 shall be returned.

3138 **Table 43 – HTTP status codes for failing POST (deliver an indication)**

HTTP status code	HTTP status text	Generic operations error ID	Generic operations error title
400	Bad Request	WIPG0205	Missing input parameter
400	Bad Request	WIPG0206	Duplicate input parameter
400	Bad Request	WIPG0207	Unknown input parameter
400	Bad Request	WIPG0208	Incompatible input parameter type
400	Bad Request	WIPG0249	Invalid input parameter value

3139 **Example HTTP conversation (using JSON as defined in DSP0211):**

3140 Request (if type information is included in the request and accepted to be included in an error response):

```
3141  POST /destinations/dest1/indications HTTP/1.1
3142  Host: listener.acme.com:5988
3143  Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
3144  Content-Length: XXX
3145  Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
3146  X-CIMRS-Version: 2.0.1
3147
3148  {
3149    "kind": "indicationdeliveryrequest",
3150    "self": "/destinations/dest1/indications",
3151    "indication": {
3152      "kind": "instance",
3153      // self is omitted for embedded instances
3154      // namespace is omitted for embedded instances
3155      "classname": "ACME_AlertIndication",
3156      "properties": {
3157        "AlertType": {"type": "uint16", "value": 4},
```

```
3158            "PerceivedSeverity": {"type": "uint16", "value": 5},
3159            "ProbableCause": {"type": "uint16", "value": 42},
3160            "Message": {"type": "string",
3161                        "value": "BOND0007: Some error happened, rc=23."},
3162            "MessageArguments": {"type": "string", "array": True, "value": [ "23" ]},
3163            "MessageID": {"type": "string", "value": "BOND0007"},
3164            "OwningEntity": {"type": "string", "value": "ACME"}
3165          }
3166       }
3167    }
```

3168   Response:

```
3169    HTTP/1.1 204 No Content
3170    Date: Thu, 30 Oct 2014 15:03:00 GMT
3171    X-CIMRS-Version: 2.0.0
```

3172   7.17 CIM-RS resources to be exposed (informative)

3173   This subclause informatively summarizes which resources servers and listeners need to expose.

3174   7.17.1 Resources exposed by a server

3175   For each namespace, the following resources are exposed by a server:

3176   •   Class collection resource (see 7.11)

3177   •   Qualifier type collection resource (see 7.15)

3178   For each qualifier type in each namespace, the following resources are exposed by a server:

3179   •   Qualifier type resource (see 7.14)

3180   For each class in each namespace, the following resources are exposed by a server:

3181   •   Class resource (see 7.10)

3182   •   Class associator collection resource (see 7.12)

3183   •   Class reference collection resource (see 7.13)

3184   •   Instance collection resource (see 7.6)

3185   For each instance (including association instances) in each namespace, the following resources are
3186   exposed by a server:

3187   •   Instance resource (see 7.5)

3188   •   Instance associator collection resource (see 7.7)

3189   •   Instance reference collection resource (see 7.8)

3190   For each open paged instance collection, the following resources are exposed by a server:

3191   •   Instance collection page resource (see 7.9)

3192   In addition, resources that support query parameters have variations based upon their query parameter
3193   values.

3194 **7.17.2 Resources exposed by a listener**

3195 For each listener destination, the following resources are exposed by a listener:

3196   • Listener indication delivery resource (see 7.16)

3197 **7.18 Other typical WBEM protocol functionality (informative)**

3198 Certain functionality that is typical for a WBEM protocol or for systems management protocols in general
3199 does not have specific operations defined in the CIM-RS protocol, but can be performed by using other
3200 operations defined in the CIM-RS protocol, or discovery protocols, or the functionality of model-defined
3201 management interfaces accessible through the CIM-RS protocol. This subclause informatively describes
3202 how a number of such functionalities can be performed.

3203 **7.18.1 Server discovery**

3204 WBEM servers can be discovered as described in clause 10.

3205 **7.18.2 Namespace discovery**

3206 The set of namespaces of a server can be discovered by clients using any of these approaches:

3207   • From the Namespaces attribute of the SLP discovery data. For details, see clause 10.

3208   • From instances of the class CIM_Namespace in the Interop namespace. See the Profile
3209   Registration Profile (DSP1033) for the concept and names of the Interop namespace. See 7.6.4
3210   for enumerating instances of a class. Note that the use of this class for representing CIM
3211   namespaces is not covered by any DMTF standard, but is commonly implemented by WBEM
3212   servers.

3213   • The WBEM Server Profile (DSP1092) describes how namespaces can be discovered. In short,
3214   namespaces are represented by instances of class CIM_WBEMServerNamespace in the
3215   Interop namespace. See 7.6.4 for enumerating instances of a class. This is the standards-
3216   based alternative to the previous approach, but is not yet commonly implemented by WBEM
3217   servers at the time of the publishing of version 2.0.0 of this document.

3218 **7.18.3 Registered profile discovery**

3219 The Profile Registration Profile (DSP1033) describes how to discover the management profiles to which a
3220 server advertises conformance, and from there, all further resources that are part of the functionality of a
3221 management profile. In short, the management profiles to which a server advertises conformance can be
3222 discovered by enumerating instances of class CIM_RegisteredProfile in the Interop namespace (see
3223 7.6.4 for enumerating instances of a class).

3224 **7.18.4 Schema inspection**

3225 The schema definition (that is, class declarations and qualifier type declarations) including its meta-data
3226 in the form of qualifiers is directly accessible via the class and qualifier operations of the CIM-RS protocol
3227 (see 7.10 and following subclauses).

3228 **7.18.5 Association traversal**

3229 The CIM-RS protocol supports traversal of associations in a way consistent to generic operations (see
3230 DSP0223). For details on association traversal operations between instances, see 7.7 and 7.8. For
3231 details on association traversal operations at the class level, see 7.12 and 7.13.

3232 **7.18.6 Indication subscription**

3233 The CIM-RS protocol defines the HTTP POST method on listener indication delivery resources (see
3234 7.16.3) for the delivery of indications (that is, event notifications). However, it does not define any specific
3235 operations for performing other indication-related functions such as subscribing for indications, retrieving
3236 and managing indication filters and filter collections, or retrieving and managing listener destinations or
3237 indication services.

3238 Consistent with other WBEM protocols, the CIM-RS protocol leaves the definition of such functionality to a
3239 model-defined management interface, such as the *Indications Profile* (DSP1054).

3240 # 8   HTTP usage

3241 ## 8.1   General requirements

3242 WBEM clients, servers, and listeners may support the use of HTTP for the CIM-RS protocol. The
3243 following applies if HTTP is supported:

3244   •   Version 1.1 of HTTP shall be supported as defined in RFC2616.

3245   •   Version 1.0 or earlier of HTTP shall not be supported.

3246 WBEM clients, servers, and listeners shall support the use of HTTPS for the CIM-RS protocol. The
3247 following applies:

3248   •   HTTPS shall be supported as defined in RFC2818.

3249   •   Within HTTPS, version 1.1 of HTTP shall be supported as defined in RFC2616.

3250 NOTE   HTTPS should not be confused with Secure HTTP defined in RFC2660.

3251 ## 8.2   Authentication requirements

3252 This subclause describes requirements and considerations for authentication between clients, servers,
3253 and listeners. Specifically, authentication happens from clients to servers for operation messages, and
3254 from servers to listeners for indication delivery messages.

3255 ### 8.2.1   Operating without authentication

3256 WBEM clients, servers, and listeners may support operating without the use of authentication.

3257 This may be acceptable in environments such as physically isolated networks or between components on
3258 the same operating system.

3259 ### 8.2.2   HTTP basic authentication

3260 HTTP basic authentication provides a rudimentary level of authentication, with the major weakness that
3261 the client password is part of the HTTP headers in unencrypted form.

3262 WBEM clients, servers, and listeners may support HTTP basic authentication as defined in RFC2617.

3263 HTTP basic authentication may be acceptable in environments such as physically isolated networks,
3264 between components on the same operating system, or when the messages are encrypted by using
3265 HTTPS.

3266 ### 8.2.3   HTTP digest authentication

3267 HTTP digest authentication verifies that both parties share a common secret without having to send that
3268 secret in the clear. Thus, it is more secure than HTTP basic authentication.

3269 WBEM clients, servers, and listeners should support HTTP digest authentication as defined in RFC2617.

8.2.4 Other authentication mechanisms

3270

3271 WBEM clients, servers, and listeners may support authentication mechanisms not covered by RFC2617.
3272 One example of such a mechanism is public key certificates as defined in X.509.

8.3 Message encryption requirements

3273

3274 Encryption of HTTP messages can be supported by the use of HTTPS and its secure sockets layer.

3275 It is important to understand that authentication and encryption of messages are separate issues:
3276 Encryption of messages requires the use of HTTPS, while the authentication mechanisms defined in 8.2
3277 can be used with both HTTP and HTTPS.

3278 The following requirements apply to clients, servers, and listeners regarding the secure sockets layer
3279 used with HTTPS:

3280 • TLS 1.0 (also known as SSL 3.1) as defined in RFC2246 shall be supported. Note that TLS 1.0
3281 implementations may be vulnerable when using CBC cipher suites

3282 • TLS 1.1 as defined in RFC4346 should be supported

3283 • TLS 1.2 as defined in RFC5246 should be supported

3284 • SSL 2.0 or SSL 3.0 shall not be supported because of known security issues in these versions

3285 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
3286 time of publication of this standard, it is expected that support for TLS 1.1 and TLS 1.2 is still not
3287 pervasive; therefore TLS 1.0 has been chosen as a minimum despite its known security issues.

3288 RFC5246 describes in Appendix E "Backward Compatibility" how the secure sockets layer can be
3289 negotiated.

3290 The following requirements apply to clients, servers, and listeners regarding the cipher suites used with
3291 HTTPS:

3292 • The TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value 0x0013)
3293 shall be supported when using TLS 1.0. Note that RFC2246 defines this cipher suite to be
3294 mandatory for TLS 1.0

3295 • The TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value 0x000A) shall
3296 be supported when using TLS 1.1. Note that RFC4346 defines this cipher suite to be mandatory
3297 for TLS 1.1

3298 • The TLS_RSA_WITH_AES_128_CBC_SHA cipher suite (hexadecimal value 0x002F) shall be
3299 supported when using TLS 1.2. Note that RFC5246 defines this cipher suite to be mandatory for
3300 TLS 1.2

3301 • The TLS_RSA_WITH_AES_128_CBC_SHA256 cipher suite (hexadecimal value 0x003C)
3302 should be supported when using TLS 1.2, in order to meet the transition to a security strength of
3303 112 bits (guidance is provided in NIST Special Publication 800-57 [NIST 800-57] and NIST
3304 Special Publication 800-131A [NIST 800-131A])

3305 • Any additional cipher suites may be supported

8.4 HTTP header fields

3306

3307 This subclause describes the use of HTTP header fields within the CIM-RS protocol, and it defines
3308 extension-header fields specific to the CIM-RS protocol.

3309   Any rules for processing header fields defined in RFC2616 apply, particularly regarding whitespace
3310   stripping, line continuation, multiple occurrences of headers, and case insensitive treatment of field
3311   names.

8.4.1   Accept

3313   The rules for the Accept request-header field defined in RFC2616 apply. This subclause defines
3314   additional constraints on its use.

3315   The Accept header field shall be provided on the request message of every operation. The reason is that
3316   any operation may fail and the failure response will include an ErrorResponse payload element (see
3317   7.3.5).

3318   The Accept header field shall specify media types identifying CIM-RS payload representations (including
3319   version) that are supported by the client.

3320   The use of media ranges (that is, the asterisk character "*") in the type or subtype fields of the media type
3321   is not permitted in the CIM-RS protocol.

3322   NOTE   RFC2616 permits the use of media ranges for the Accept header field. However, with the envisioned
3323   combinations of type and subtype values for CIM-RS, wildcarding based on type and subtype is not meaningful.

3324   If implemented, the "q" accept parameter shall be interpreted as a preference; interpreting it as a quality
3325   does not make sense for the CIM-RS protocol. Clients may provide the "q" accept parameter. Servers
3326   should implement the "q" accept parameter; if not implemented, it shall be tolerated if provided.

3327   NOTE   RFC2616 does not specify recommendations for implementing the "q" accept parameter.

3328   NOTE   RFC2616 distinguishes between general media type parameters (such as "version"), and accept parameters
3329   (such as "q"); the latter can be used only in the Accept header field, while general media type parameters can be
3330   considered part of the media type definition.

3331   Additional accept parameters (that is, beyond "q") are not permitted to be used in the Accept header field.
3332   For future extensibility, servers shall tolerate and ignore unknown additional accept parameters.

3333   A server shall use one of the payload representations and versions identified in the Accept header field
3334   for the response payload, considering the "q" accept parameter if implemented.

3335   The payload representation version specified in the media type (see 9.1) shall be interpreted by the
3336   server as follows:

3337   •   The update version is optional to be included. If an update version is included, it specifies the
3338   lowest acceptable update version (within the specified major version and acceptable minor
3339   versions); higher update versions shall be acceptable in addition. If no update version is
3340   included, the server shall assume a default of 0; that is, any update version is acceptable (within
3341   the specified major version and acceptable minor versions).

3342   •   The minor version is required to be included and specifies the only acceptable minor version.

3343   •   The major version is required to be included and specifies the only acceptable major version.

3344   NOTE   These rules follow the usual DMTF convention for referencing versions: Update versions newer than the one
3345   specified are selected automatically if available, but newer minor (and of course, major) versions are not selected
3346   automatically.

3347   If none of the payload representations identified in the Accept header field is supported by the server, it
3348   shall return HTTP status code 406 "not acceptable".

3349   NOTE   RFC2616 only recommends returning HTTP status code 406 "not acceptable" in this case, but it does not
3350   require it.

3351   If no Accept header field is provided, servers may use any valid payload representation and version for
3352   the response payload.

3353 Within the constraints defined in this subclause, the payload representations specified in the Accept
3354 header field and the payload representations used in the response may change over time, even between
3355 the same combination of client and server. This implies that a server needs to evaluate the Accept header
3356 field (if present) on every request, even when the request is originated from the same client as before.

3357 The following example assumes a JSON-based payload representation identified by
3358 "`application/json`" and an XML-based payload representation identified by "`text/xml`". Actual
3359 payload representations may define different media types.

3360 Example:

```
3361    Accept: application/json; version=2.0,
3362            application/json;version=2.0.1; q=0.5,
3363            text/xml; version=2.0;q=0.2
```

3364 In this example, the value of the Accept header field is distributed over multiple lines. The client
3365 expresses a preference for version 2.0.x (x>=0) of the (assumed) JSON-based payload representation
3366 (by means of the default value of 1 for the "q" parameter), if that representation version is not available,
3367 then for version 2.0.x (x>=1) of the JSON-based representation, if that is not available then for version
3368 2.0.x (x>=0) of the (assumed) XML-based representation.

3369 8.4.2   Content-Type

3370 The rules for the Content-Type entity-header field defined in [RFC2616](#) apply. This subclause defines
3371 additional constraints on its use.

3372 As defined in [RFC2616](#), the Content-Type entity-header field shall be provided on the request message
3373 of any operation that passes a request payload and on the response message of any operation that
3374 returns a response payload.

3375 The Content-Type entity-header field shall specify the media type identifying the CIM-RS payload
3376 representation and version that is used for the content of the entity body. The payload representation
3377 version indicated by the media type shall include the major, minor and update version indicators.

3378 8.4.3   X-CIMRS-Version

3379 The CIM-RS protocol version is the version of this document, without any draft level.

3380 The X-CIMRS-Version extension-header field shall identify the CIM-RS protocol version to which the
3381 request or response conforms, using the following format for its field value (defined in ABNF):

```
3382    X-CIMRS-Version-value = M "." N "." U
```

3383 where `M` is the major version indicator, `N` is the minor version indicator, and `U` is the update version
3384 indicator within the version. Each of these version indicator strings shall be a decimal representation of
3385 the corresponding version indicator number without leading zeros. Note that each indicator version string
3386 may include more than a single decimal digit.

3387 The X-CIMRS-Version extension-header field shall be included in any request and in any response.

3388 Example:

```
3389    X-CIMRS-Version: 2.0.0
```

3390 9 Payload representation

3391 CIM-RS payload representation specifications define how the abstract payload elements defined in this
3392 document are encoded in the entity body of the HTTP messages used by the CIM-RS protocol. Such an
3393 encoding format is termed a "*payload representation*" in this document.

3394 This clause defines requirements for payload representation specifications and for implementations of the
3395 CIM-RS protocol that are related to payload representations.

3396 9.1 Internet media types

3397 The CIM-RS protocol uses Internet media types for identifying the payload representation of its abstract
3398 payload elements. This subclause defines requirements related to media types used for the CIM-RS
3399 protocol.

3400 Each CIM-RS payload representation specification shall define a media type as defined in RFC6838 and
3401 RFC6839 that uniquely identifies its payload representation within the set of payload representations
3402 listed in Table 44, and that identifies the version of the payload representation (typically by using a media
3403 type parameter such as "version").

3404 It is recommended that any such media types be registered with IANA.

3405 9.2 Payload element representations

3406 CIM-RS payload representation specifications shall define a representation for each payload element
3407 listed in Table 4.

3408 The representations of these payload elements should be designed such that they can represent
3409 elements from any valid model without introducing restrictions, and such that there is no need to extend
3410 the payload representation specification if the model gets extended.

3411 Attributes of the payload elements defined in this document may be represented in any way in the
3412 payload representation. The attribute names stated in the descriptions of the payload elements in clause
3413 7 do not need to be retained in the payload representation. The payload data types stated in Table 5 do
3414 not need to correspond 1:1 to data types the representation format may use, as long as the value range
3415 of the attribute values can be correctly represented without any restrictions or loss of information.

3416 For example, in a JSON representation of an Instance payload element (see 7.5.2), all of the following
3417 options would be valid for representing the "self" attribute for resource identifier "/machine/1234":

3418   • as a JSON attribute with the same name as the attribute of the abstract payload element:

```
3419  {
3420    "self": "/root%2Fcimv2/classes/ACME_ComputerSystem/instances/sys11",
3421
3422    "self": {
3423      "href": "/root……",
3424      "classname": "..",
3425      "namespace": "..",
3426      "keys": { "key1": <like any property value>, …}
3427    }
3428    . . .
3429  }
```

3430 • as a JSON attribute with a different name as the attribute of the abstract payload element:

```
3431  {
3432    "this": "/root%2Fcimv2/classes/ACME_ComputerSystem/instances/sys11",
3433    . . .
3434  }
```

3435 • as an entry in a JSON array for links following the rel/href approach:

```
3436  {
3437    "links": [
3438      { "rel": "self",
3439        "href": "/root%2Fcimv2/classes/ACME_ComputerSystem/instances/sys11" },
3440       . . .
3441    },
3442    . . .
3443  }
```

9.3 Payload representations

3444

3445 Table 44 lists known payload representations for this major version of the CIM-RS protocol and
3446 requirements to implement them; payload representations not listed in Table 44 may be implemented in
3447 addition.

3448 This table will be kept up to date in future versions of this document to include known payload
3449 representations, in order to provide a basis on which the media type can be kept unique.

3450   **Table 44 – CIM-RS payload representations**

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

10 Discovery requirements

3451

3452 The CIM-RS protocol has the following requirements related to discovery protocols:

3453 WBEM servers should implement the SLP discovery protocol, supporting the provisions set forth in
3454 DSP0205, and the SLP template defined in DSP0206.

3455 The CIM-RS protocol has no requirements for supporting the discovery of listeners.

11 Version compatibility

3456

3457 This clause defines the rules for version compatibility between WBEM clients and servers.

3458 Since HTTP is session-less, the general principle for determining version compatibility in the CIM-RS
3459 protocol is that the version for the relevant layers of the CIM-RS protocol is included in all protocol
3460 messages, allowing the receiving participant to determine whether it is able to support that version.

3461 The general principle for backwards compatibility (as further detailed in this clause) is that servers are
3462 backwards compatible to clients; that is, servers of a particular version work with "older" versions of
3463 clients.

3464 Version compatibility for the CIM-RS protocol is defined for the following protocol layers:

3465 • HTTP protocol (see 11.1)

3466 • CIM-RS protocol (see 11.2)

3467 • CIM-RS payload representation (see 11.3)

3468 A client and a server are version-compatible with each other only if they are compatible at each of these
3469 three protocol layers.

11.1 HTTP protocol version compatibility

3471 As defined in RFC2616, every HTTP request and every HTTP response shall indicate the HTTP protocol
3472 version to which the message format conforms.

3473 Since the CIM-RS protocol requires support for HTTP 1.1 (see 8.1), the backward compatibility rules for
3474 supporting HTTP 1.0 and HTTP 0.9 as defined in section 19.6 (Compatibility with Previous Versions) of
3475 RFC2616 do not need to be followed in order to conform to the CIM-RS protocol.

3476 At this point, there is no HTTP version higher than 1.1 defined.

11.2 CIM-RS protocol version compatibility

3478 As defined in 8.4.3, every HTTP request and every HTTP response in the CIM-RS protocol shall indicate
3479 the CIM-RS protocol version to which the request or response conforms, by including the X-CIMRS-
3480 Version extension-header field. As defined in 8.4.3, the X-CIMRS-Version extension-header field
3481 identifies major, minor and update version of the CIM-RS protocol.

3482 A client and a server are compatible w.r.t. the CIM-RS protocol version only if the following condition is
3483 satisfied:

3484 • the major version of the server is equal to the major version of the client, and the minor version
3485 of the server is equal to or larger than the minor version of the client.

3486 The update version is not considered in this rule because new update versions (within the same major
3487 and minor version) are not supposed to introduce new functionality, so this rule allows clients and servers
3488 to be upgraded to conform to new update versions of the CIM-RS protocol independently of each other.

11.3 CIM-RS payload representation version compatibility

3490 As defined in 9.1, the CIM-RS payload representation is identified using a media type whose "version"
3491 parameter identifies its major, minor and update version.

3492 A client and a server are compatible w.r.t. the version of a particular payload representation only if the
3493 following condition is satisfied:

3494 • the major version of the server is equal to the major version of the client, and the minor version
3495 of the server is equal to or larger than the minor version of the client.

3496 The update version is not considered in this rule because new update versions (within the same major
3497 and minor version) are not supposed to introduce new functionality, so this rule allows clients and servers
3498 to be upgraded to conform to new update versions of the payload representation independently of each
3499 other.

12 Conformance

3501 This clause defines the criteria for WBEM clients, servers, and listeners to implement the CIM-RS
3502 protocol conformant to this document.

3503   WBEM clients, servers, and listeners implement the CIM-RS protocol conformant to this document only if
3504   they satisfy all provisions set out in this document.

3505   The terms client, server, and listener in this document refer to clients, servers, and listeners that are
3506   conformant to this document, without explicitly mentioning that.

3507	**ANNEX A**
3508	**(normative)**
3509	
3510	**Common ABNF rules**

3511   This annex defines common ABNF rules used throughout this document.

```
3512  nonZeroDecimalDigit = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"

3513  decimalDigit = "0" / nonZeroDecimalDigit

3514  leadingZeros = 1*"0"

3515  positiveDecimalInteger = [leadingZeros] nonZeroDecimalDigit *decimalDigit

3516  nonNegativeDecimalInteger = [leadingZeros] ( "0" / nonZeroDecimalDigit *decimalDigit )
```

3517

ANNEX B
(normative)

3520

3521 **Mapping CIM-RS to generic operations**

3522 This annex describes how CIM-RS operations shall be mapped to generic operations (see DSP0223).
3523 This mapping is useful when implementing the CIM-RS protocol in WBEM servers and listeners that
3524 internally support the semantics of generic operations.

3525 **B.1   Query parameters**

3526 Most of the CIM-RS query parameters (see 6.6) can be used with multiple CIM-RS operations. Likewise,
3527 many generic operations input parameters are common between multiple generic operations, and are
3528 used consistently across those operations. With minor exceptions, the usage of any particular CIM-RS
3529 query parameter can be mapped directly to specific generic operation parameters, regardless of the CIM-
3530 RS operation with which it is used.

3531 Table 45 defines the mapping of CIM-RS query parameters to generic operations input parameters.

3532   **Table 45 – Mapping of CIM-RS query parameters to generic operations input parameters**

CIM-RS Query Parameter	Description	Generic Operations Parameter	Mapping
$associatedclass	see 6.6.1	AssociatedClassName	Directly equivalent
$associatedrole	see 6.6.2	AssociatedRoleName	Directly equivalent
$associationclass	see 6.6.3	AssociationClassName	Directly equivalent
$class	see 6.6.4	N/A	See the individual operation/resource mappings in this annex
$continueonerror	see 6.6.5	ContinueOnError	Directly equivalent
$filter	see 6.6.6	FilterQueryString	Directly equivalent
$filterlanguage	see 6.6.7	FilterQueryLanguage	Directly equivalent
$max	see 6.6.8	MaxObjectCount	Directly equivalent
$pagingtimeout	see 6.6.9	OperationTimeout	Directly equivalent
$properties	see 6.6.10	IncludedProperties	Directly equivalent for instance operations; Always unspecified for class operations (see C.2)
N/A	N/A	IncludeInheritedElements	Always set to TRUE (see C.2)
$sourcerole	see 6.6.11	SourceRoleName	Directly equivalent
$subclasses	see 6.6.12	IncludeSubclasses	Directly equivalent
$qualifiers	see 6.6.13	IncludeQualifiers	Directly equivalent

3533 **B.2   Server operations**

3534 This subclause describes a server's decision tree for how incoming CIM-RS operations shall be analyzed,
3535 identified, and mapped to generic operations. The server can determine the generic operation based on
3536 the HTTP method and the target resource type.

3537 The target resource type can be identified from the format of the path component of the target resource
3538 identifier, as shown in Table 46.

3539 **Table 46 – Identifying the server's target resource type from the target resource identifier**

Path Component of Target Resource Identifier	Target Resource Type
`/{nsname}/qualifiertypes`	Qualifier type collection
`/{nsname}/qualifiertypes/{qualifiername}`	Qualifier type
`/{nsname}/classes`	Class collection
`/{nsname}/classes/{classname}`	Class
`/{nsname}/classes/{classname}/associators`	Class associator collection
`/{nsname}/classes/{classname}/references`	Class reference collection
`/{nsname}/classes/{classname}/instances`	Instance collection (of class)
`/{nsname}/classes/{classname}/instances/{keys}`	Instance
`/{nsname}/classes/{classname}/instances/{keys}/associators`	Instance associator collection
`/{nsname}/classes/{classname}/instances/{keys}/references`	Instance reference collection
server-implementation-specific format	Instance collection page

3540 The generic operation(s) that shall be invoked for each combination of HTTP method and resource type
3541 are shown in Table 47. The query parameters shall be mapped to generic operation parameters as
3542 described in Table 45; column "Generic Operation Parameters" in Table 47 lists additional constraints on
3543 generic operation parameters.

3544 **Table 47 – Mapping CIM-RS server operations to generic operations**

HTTP Method	Target Resource Type	Generic Operation	Generic Operation Parameters	Description
GET	Instance	GetInstance	InstancePath is set from target resource identifier	see 7.5.5
PUT	Instance	ModifyInstance	InstancePath is set from target resource identifier; ModifiedInstance is set from payload.	see 7.5.6
DELETE	Instance	DeleteInstance	InstancePath is set from target resource identifier	see 7.5.7
POST	Instance	InvokeMethod, InvokeStaticMethod on instance	InstancePath is set from target resource identifier; MethodName and InParmValues are set from payload.	see 7.5.8
POST	Instance collection	CreateInstance	ClassPath is set from target resource identifier; NewInstance is set from payload.	see 7.6.3

HTTP Method	Target Resource Type	Generic Operation	Generic Operation Parameters	Description
GET	Instance collection	OpenEnumerateInstances	EnumClassPath is set from target resource identifier. On return, EndOfSequence determines whether the "next" attribute is set, and EnumerationContext is used to construct its value.	see 7.6.4
GET	Instance associator collection	OpenAssociators	SourceInstancePath is set from target resource identifier. On return, EndOfSequence determines whether the "next" attribute is set, and EnumerationContext is used to construct its value.	see 7.7.2
GET	Instance reference collection	OpenReferences	SourceInstancePath is set from target resource identifier. On return, EndOfSequence determines whether the "next" attribute is set, and EnumerationContext is used to construct its value.	see 7.8.2
GET	Instance collection page	PullInstancesWithPath	NamespacePath and EnumerationContext are set from target resource identifier	see 7.9.2
DELETE	Instance collection page	CloseEnumeration	NamespacePath and EnumerationContext are set from target resource identifier	see 7.9.3
GET	Class	GetClass	ClassPath is set from target resource identifier; IncludedProperties	see 7.10.3
PUT	Class	ModifyClass	ClassPath is set from target resource identifier; ModifiedClass is set from payload	see 7.10.4
DELETE	Class	DeleteClass	ClassPath is set from target resource identifier	see 7.10.5
POST	Class	InvokeStaticMethod on class	ClassPath is set from target resource identifier; MethodName and InParmValues are set from payload.	see 7.10.6
POST	Class collection	CreateClass	NamespacePath is set from target resource identifier; NewClass is set from payload.	see 7.11.3
GET	Class collection	EnumerateClasses	NamespacePath and ClassName are set from target resource identifier	see 7.11.4
GET	Class associator collection	AssociatorClasses	ClassPath is set from target resource identifier	see 7.12.2
GET	Class reference collection	ReferenceClasses	ClassPath is set from target resource identifier	see 7.13.2

HTTP Method	Target Resource Type	Generic Operation	Generic Operation Parameters	Description
GET	Qualifier type	GetQualifierType	QualifierTypePath is set from target resource identifier	see 7.14.3
PUT	Qualifier type	ModifyQualifierType	QualifierTypePath is set from target resource identifier; ModifiedQualifierType is set from payload.	see 7.14.4
DELETE	Qualifier type	DeleteQualifierType	QualifierTypePath is set from target resource identifier	see 7.14.5
POST	Qualifier type collection	CreateQualifierType	NamespacePath is set from target resource identifier; NewQualifierType is set from payload.	see 7.15.3
GET	Qualifier type collection	EnumerateQualifierTypes	NamespacePath is set from target resource identifier	see 7.15.4

B.3 Listener operations

3545
3546 This subclause describes a listener's decision tree for how incoming CIM-RS listener operations shall be
3547 analyzed, identified, and mapped to generic listener operations.

3548 The listener can determine the generic operation based on the HTTP method and the target resource
3549 type.

3550 The target resource type can be identified from the format of the path component of the target resource
3551 identifier, as shown in Table 48.

3552 **Table 48 – Identifying the listener's target resource type from the target resource identifier**

Path Component of Target Resource Identifier	Target Resource Type
/destinations/{destname}/indications	Listener indication delivery

3553 The generic operation(s) that should be invoked for each combination of HTTP method and resource type
3554 are shown in Table 49. The query parameters are mapped to generic operation parameters as described
3555 in Table 45; column "Generic Operation Parameters" in Table 49 lists additional constraints on generic
3556 operation parameters.

3557 **Table 49 – Mapping CIM-RS listener operations to generic operations**

HTTP Method	Target Resource Type	Generic Operation	Generic Operation Parameters	Description
POST	Listener indication delivery	DeliverIndication	ListenerDestination is set from target resource identifier; Indication is set from payload.	see 7.16.3

3558   **ANNEX C**
3559   **(normative)**

3560

3561   **Mapping generic operations to CIM-RS**

3562   This annex describes how generic operations (see [DSP0223](#)) are mapped to CIM-RS operations. This
3563   mapping is provided primarily to describe how the CIM-RS protocol conforms to generic operations. This
3564   mapping is also useful for translating operation requirements defined in management profiles that are
3565   stated in terms of generic operations, into CIM-RS operations.

3566   ## C.1   Conformance

3567   CIM-RS does not satisfy all conformance requirements defined in generic operations ([DSP0223](#)). As a
3568   result, CIM-RS is not a conforming WBEM protocol. The remaining subclauses in this annex provide
3569   details.

3570   ## C.2   Support of optional generic operations features

3571   This subclause describes how CIM-RS supports optional features defined in generic operations.

3572   - CIM-RS does not support the exclusion of all inherited properties and methods with one
3573   parameter when retrieving classes (that is, the equivalent of generic operation parameter
3574   `IncludeInheritedElements=False`).

3575   - CIM-RS supports the inclusion of specific properties when retrieving instances (that is, the
3576   equivalent of generic operation parameter `IncludedProperties`)

3577   - CIM-RS supports the specification of initial property values when creating an instance (that is,
3578   the equivalent of generic operation parameter `NewInstance`)

3579   - CIM-RS supports error handling by means of returning DMTF standard messages (also known
3580   as "extended error handling")

3581   - CIM-RS supports filter queries in pulled instance operations (that is, the equivalent of generic
3582   operation parameter `FilterQueryString`). The DMTF *Filter Query Language* (see
3583   [DSP0212](#)) is required to be supported as a query language. Other query languages are not
3584   currently supported with CIM-RS.

3585   - CIM-RS supports client side control of continuation on error for pulled instance enumeration
3586   operations (that is, the equivalent of generic operation parameter `ContinueOnError`)

3587   ## C.3   Operations

3588   This subclause describes how the generic operations are supported in CIM-RS.

3589   ### C.3.1   Server operations

3590   The generic server operations listed in Table 47 are supported as described there.

3591   Table 50 lists generic server operations that are not supported in CIM-RS. These operations are the
3592   reason CIM-RS does not conform to [DSP0223:](#)

3593 **Table 50 – Generic server operations not supported in CIM-RS**

Generic Operation	Remarks
OpenQueryInstances	
PullInstances	

3594 **C.3.2   Listener operations**

3595   The generic listener operations listed in Table 49 are supported as described there.

3596   **ANNEX D**
3597   **(informative)**

3598

3599   **Change log**

3600

Version	Date	Description
1.0.0	2013-01-24	
1.0.1	2014-02-11	
2.0.0	2015-03-06	Released as a DMTF Standard, with the following changes compared to 1.0.1: • Added support for classes and qualifier types • Well-defined, non-opaque resource URIs • Substantial changes to method invocation • Eliminated special enumeration, method invocation, and entry point resources • Redefined navigation between instances such that it reflects the generic association traversal operations 1:1 • Specified HTTP status codes for each method • Upgraded to version 2.0 of generic operations (DSP0223)

3601

Bibliography

3602   This annex contains a list of informative references for this document.

3603   DMTF DSP0200, *CIM Operations over HTTP 1.3*,
3604   http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf

3605   DMTF DSP1001, *Management Profile Specification Usage Guide 1.1*,
3606   http://www.dmtf.org/standards/published_documents/DSP1001_1.1.pdf

3607   DMTF DSP1033, *Profile Registration Profile 1.1*,
3608   http://www.dmtf.org/standards/published_documents/DSP1033_1.1.pdf

3609   DMTF DSP1054, *Indications Profile 1.2*,
3610   http://www.dmtf.org/standards/published_documents/DSP1054_1.2.pdf

3611   DMTF DSP1092, *WBEM Server Profile 1.0*,
3612   http://www.dmtf.org/standards/published_documents/DSP1092_1.0.pdf

3613   DMTF DSP2032, *CIM-RS White Paper 1.0*,
3614   http://www.dmtf.org/standards/published_documents/DSP2032_1.0.pdf

3615   IETF RFC2608, *Service Location Protocol, Version 2*, June 1999,
3616   http://tools.ietf.org/html/rfc2608

3617   IETF RFC4648, *The Base16, Base32, and Base64 Data Encodings*, October 2006,
3618   http://tools.ietf.org/html/rfc4648

3619   IETF RFC5005, *Feed Paging and Archiving*, September 2007,
3620   http://tools.ietf.org/html/rfc5005

3621   IETF RFC7159, *The JavaScript Object Notation (JSON) Data Interchange Format*, March 2014,
3622   http://tools.ietf.org/html/rfc7159

3623   IANA Permanent Message Header Field Names,
3624   http://www.iana.org/assignments/message-headers/perm-headers.html

3625   IANA MIME Media Types,
3626   http://www.iana.org/assignments/media-types/

3627   IANA Port Number Registry
3628   http://www.iana.org/assignments/port-numbers

3629   ITU-T X.509, *Information technology – Open Systems Interconnection – The Directory: Public-key and*
3630   *attribute certificate frameworks*,
3631   http://www.itu.int/rec/T-REC-X.509/en

3632   R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, PhD thesis,
3633   University of California, Irvine, 2000,
3634   http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

3635   R. Fielding, *REST APIs must be hypertext driven*, October 2008,
3636   http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

3637   J. Holzer, *RESTful Web Services and JSON for WBEM Operations*, Master thesis, University of Applied
3638   Sciences, Konstanz, Germany, June 2009,
3639   http://mond.htwg-konstanz.de/Abschlussarbeiten/Details.aspx?id=1120

3640 A. Manes, *Rest principle: Separation of representation and resource*, March 2009,
3641 http://apsblog.burtongroup.com/2009/03/rest-principle-separation-of-representation-and-resource.html

3642 L. Richardson and S. Ruby, *RESTful Web Services*, May 2007, O'Reilly, ISBN 978-0-596-52926-0,
3643 http://www.oreilly.de/catalog/9780596529260/