Document Identifier: DSP0210

Date: 2015-03-06

Version: 2.0.0

CIM-RS Protocol

Supersedes: 1.0

Document Type: Specification

Document Class: Normative

Document Status: Published

Document Language: en-US

CONTENTS

Figures

Tables

Foreword

The CIM-RS Protocol (DSP0210) specification was prepared by the DMTF CIM-RS Working Group.

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

Acknowledgments

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

- Cornelia Davis, EMC
- Jim Davis, WS
- George Ericson, EMC
- Johannes Holzer, IBM
- Robert Kieninger, IBM
- Wojtek Kozaczynski, Microsoft
- Larry Lamers, VMware
- Andreas Maier, IBM (editor)
- Karl Schopmeyer, Inova
- Bob Tillman, EMC
- Marvin Waschke, CA Technologies

Introduction

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

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

Document conventions

Typographical conventions

The following typographical conventions are used in this document:

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

ABNF usage conventions

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

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

CIM-RS Protocol

1 Scope

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

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

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

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

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

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

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

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

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

2 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3 Terms and definitions

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

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

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

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

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

The following additional terms are used in this document.

3.1

CIM-RS payload data type

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

3.2

CIM-RS operation

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

3.3

CIM-RS payload element

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

3.4

CIM-RS payload representation

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

3.5

CIM-RS payload representation specification

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

3.6

CIM-RS protocol

the RESTful protocol defined in this document.

3.7

CIM-RS resource

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

3.8

CIM-RS resource identifier

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

3.9

HTTP basic authentication

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

3.10

HTTP content negotiation

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

3.11

HTTP digest authentication

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

3.12

HTTP entity body

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

3.13

HTTP entity-header field

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

3.14
HTTP extension-header field

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

3.15
HTTP general-header field

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

3.16
HTTP header field

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

3.17
HTTP message

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

3.18
HTTP method

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

3.19
HTTP request message

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

3.20
HTTP request-header field

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

3.21
HTTP response message

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

3.22
HTTP response-header field

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

3.23
Internet media type

a string identification for representation formats in Internet protocols. Originally defined for email attachments and termed "MIME type". Because the CIM-RS protocol is based on HTTP, it uses the 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".

3.34

unreserved character

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

4 Symbols and abbreviated terms

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

The following additional abbreviations are used in this document.

4.1

CIM-RS

CIM RESTful Services

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

4.2

HTTPS

Hyper Text Transfer Protocol Secure, as defined in RFC2818.

4.3

JSON

JavaScript Object Notation, as defined in RFC7159.

4.4

UTF-8

UCS Transformation Format 8, as defined in ISO/IEC 10646:2003.

5 Concepts

This clause defines concepts of the CIM-RS protocol.

5.1 CIM-RS protocol participants

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

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

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

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

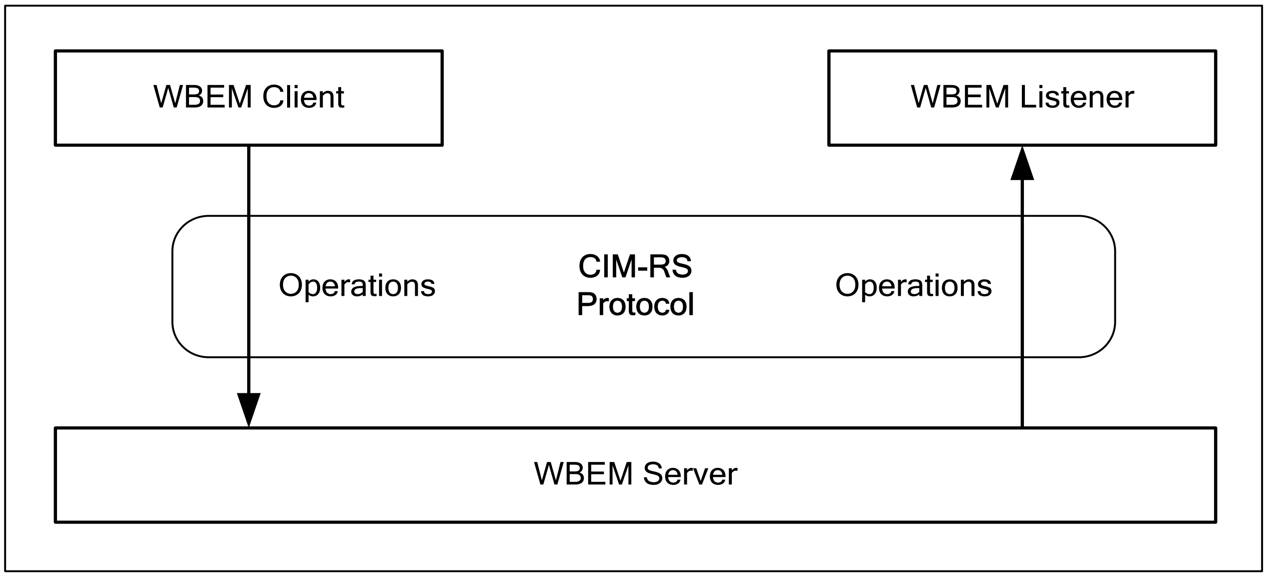

Figure 1 – Participants in the CIM-RS protocol

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

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

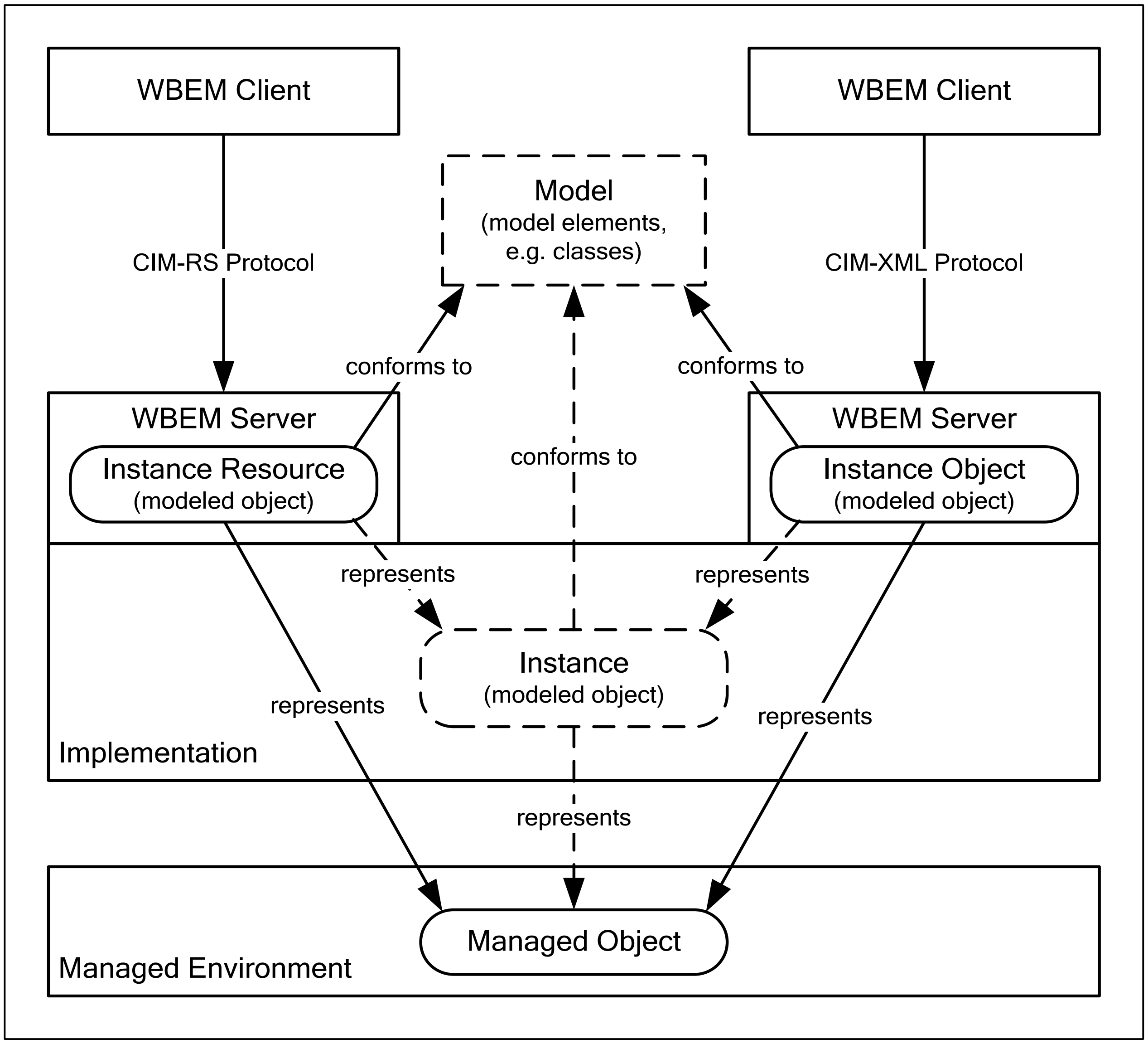

Figure 2 – Single model and multiple protocols

The separation of protocol and model at the specification level is beneficial for and targeted to infrastructures that also separate protocol and model (for example, CIMOM/provider-based WBEM servers, or WBEM client libraries). However, such a separation in the infrastructure is not required and 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).

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

5.4 Two-staged mapping approach

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

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

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

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

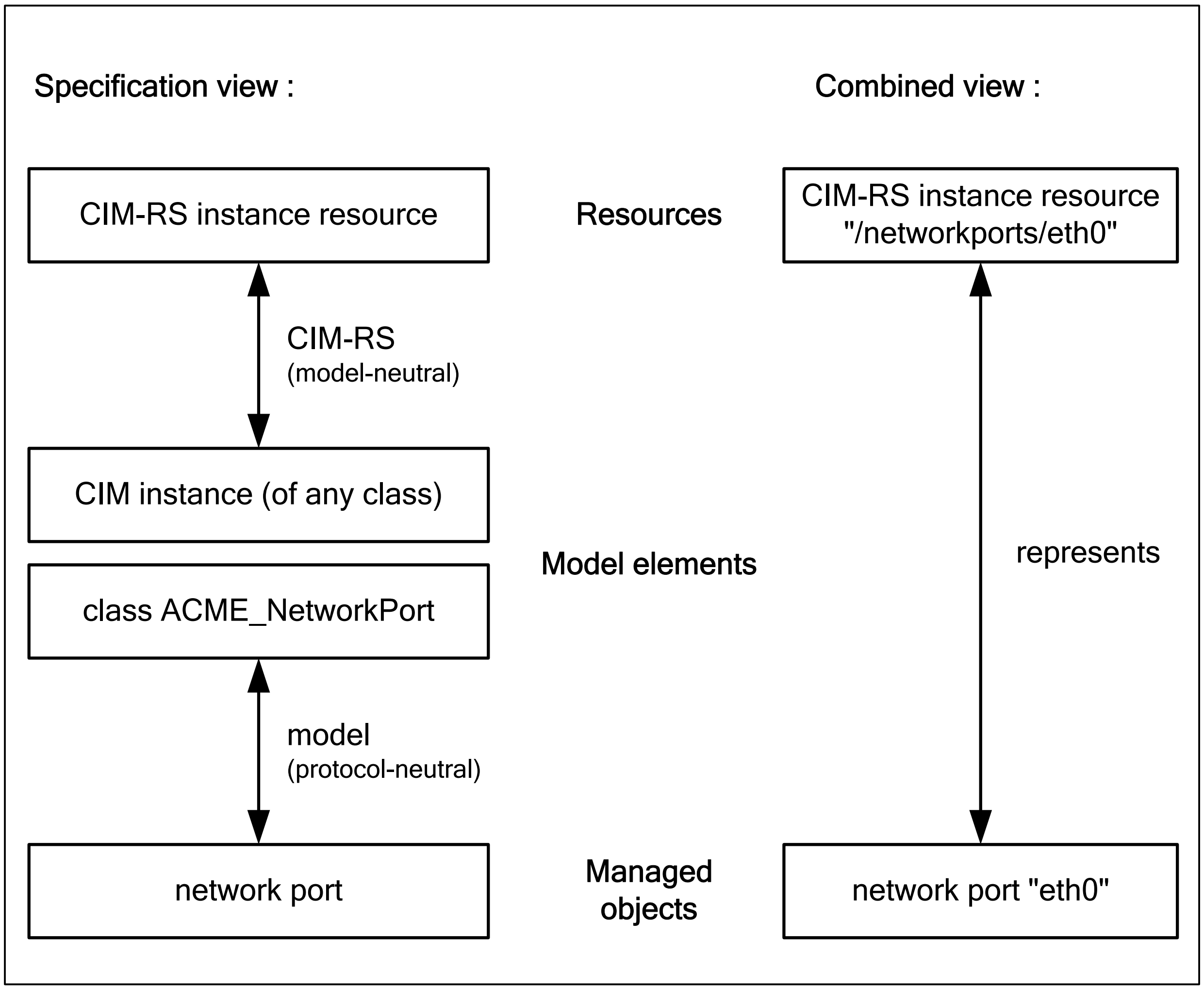

Figure 3 – Two-staged mapping approach in CIM-RS

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

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

5.5 REST architectural style supported by CIM-RS

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

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

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

  NOTE: Pulled enumeration operations as defined in DSP0223 maintain the enumeration state either on the server side or on the client side. In both approaches, the client needs to hand back and forth an opaque data item called enumeration context, which is the actual enumeration state in case of a client-maintained enumeration state, or a handle to the enumeration state in case of a server-maintained enumeration state. CIM-RS supports both of these approaches. It is possible for a server to remain stateless as far as the enumeration state goes, by implementing the client-based approach. The approach implemented by a server is not visible to a client, because the enumeration context handed back and forth is opaque to the client in both approaches.
- **Cache:** The HTTP methods used by CIM-RS are used as defined in RFC2616. As a result, they are cacheable as defined in RFC2616. This constraint is fully satisfied.

  NOTE RFC2616 defines only the result of HTTP GET methods to be cacheable.
- **Uniform interface:** The main resources represented in CIM-RS are instances or collections thereof, representing modeled objects in the managed environment. CIM-RS defines a uniform interface for creating, deleting, retrieving, replacing, and modifying these resources and thus the represented objects, based on HTTP methods. The resource identifiers used in that interface are uniformly structured. This constraint is satisfied, with the following deviation:

  Methods can be invoked in CIM-RS through the use of HTTP POST. This may be seen as a deviation from the REST architectural style, which suggests that any "method" be represented as a modification of a resource. However, DMTF experience with a REST like modeling style has shown that avoiding the use of methods is not always possible or convenient. For this reason CIM-RS supports invocation of methods.
- **Layered system:** Layering is inherent to information models that represent the objects of a managed environment, because clients only see the modeled representations and are not exposed to the actual objects. CIM-RS defines the protocol and payload representations such that it works with any model, and thus is well suited for implementations that implement a model of the managed environment independently of protocols, and one or more protocols independently of the model. CIM-RS works with HTTP intermediaries (for example, caches and proxy servers). This constraint is fully satisfied.
- **Code-On-Demand:** CIM-RS does not directly support exchanging program code between the protocol participants. This optional constraint is not satisfied.

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

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

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

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

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

6 Resource identifiers

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

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

6.1 CIM-RS resource identifier format

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

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

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

Where:

- `scheme` is defined in RFC3986 and shall in addition conform to the definitions in 6.4
- `authority` is defined in RFC3986 and shall in addition conform to the definitions in 6.5
- `path-absolute` is defined in RFC3986
- `query` is defined in RFC3986 and shall in addition conform to the definitions in 6.6

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

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

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

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

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

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

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

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

6.2 Non-opaqueness

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

Specifically, the following changes to resource identifiers are typical:

- Parsing, adding, removing or modifying any query parameters in a resource identifier
- Normalizing a resource identifier, as described in RFC3986 (for example, removing ".." and "." segments)

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

- Resource identifiers of instance collection page resources are server-implementation-specific.
- Key bindings in the resource identifiers of CIM instances may be specific to the class-specific implementation, and may not be predictable for clients.

6.3 Percent-encoding

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

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

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

  ```
  unreserved  = ALPHA / DIGIT / "-" / "." / "_" / "~"

  ALPHA = %x41-5A / %x61-7A

  DIGIT = %x30-39
  ```

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

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

  ```
  reserved  = gen-delims / sub-delims

  gen-delims    = ":" / "/" / "?" / "#" / "[" / "]" / "@"

  sub-delims    = "!" / "$" / "&" / "'" / "(" / ")"
                / "*" / "+" / "," / ";" / "="
  ```

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

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

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

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

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

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

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

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

The following examples use the minimally needed percent-encodings:

- The namespace name "root/cimv2" becomes "root%2Fcimv2" in a resource identifier, because the slash character (/) is a reserved character in resource identifiers and the subclauses on resource identifiers state that an occurrence of a slash in a namespace name is considered data.
- The class name "ACME_LogicalDevice" remains unchanged in a resource identifier, because it contains only unreserved characters.
- The (German) key property value "ÄnderungsRate" becomes "%C3%84%0AnderungsRate" in a resource identifier, because C3 84 0A are the data octets of the UTF-8 encoding of the UCS character U+00C4, which represents "Ä" (A umlaut) in normalized form. Note that usage of the UCS character sequence U+0061 U+0308 which also represents "Ä" (using the base character "A" and the combining diacritical mark ¨) is not permitted due to the requirement to use Normalization Form C.
- The string-typed value "a \"brown\" bag\n" (represented using backslash escape sequences as defined for string literals in MOF) becomes "a%20%22brown%22%20bag%0A" in a resource identifier, because the characters blank (U+0020), newline (U+000A), and double quote (U+0022) are not in the ABNF rules `reserved` and `unreserved` defined in RFC3986, and therefore need to be percent-encoded.
- The sint8-typed value -42 becomes the string "-42" in a resource identifier, because that is the string representation of an sint8-typed value defined in DSP0004, and because "-" is an unreserved character that has been chosen not to be percent-encoded in order to produce a minimally percent-encoded URI.

6.4 Scheme component

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

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

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

6.5 Authority component

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

- The `userinfo` component within `authority` shall not be specified because of security issues with specifying an unencrypted password
- The `host` component within `authority` shall be the IP (V4 or V6) address of the server, or a DNS-resolvable host name for that IP address (including `"localhost"`)
- If the `port` component within `authority` is not specified, the port number shall default to the standard port numbers for CIM-RS:
  - port number 5993 when using HTTP
  - port number 5994 when using HTTPS

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

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

6.6 Query parameters

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

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

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

Where:

- `query-parameter` is a query parameter as defined in the subclauses of this subclause
- The reserved character "&" in the literals of this ABNF rule shall be considered a delimiter for purposes of percent-encoding (see 6.3, that is, it shall not be percent-encoded).

Example:

- `$class=ACME_ComputerSystem&$subclasses=true`

  This query component specifies the query parameters `$class` with a value of `ACME_ComputerSystem` and `$subclasses` with a value of `true`
- `$properties=Name,Caption`

  This query component specifies the query parameter `$property` with a value of `Name,Caption`. The comma (,) in that value is not percent-encoded because the definition of the `$properties` query parameter (see 6.6.10) states that it is considered a delimiter.
- `$filter=Name%3D%27a%26b%27`

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

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

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

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

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

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

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

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

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

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

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

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

6.6.1 $associatedclass (associated class filter)

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

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

```
query-parameter =/ associatedclass-query-parm

associatedclass-query-parm = "$associatedclass=" class-name
```

Where:

- The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)
- `class-name` is the name of the associated class (including schema prefix). Note that CIM class names do not contain reserved characters (see 6.3 and DSP0004)

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

Examples:

(not specified)

specifies no filtering on the associated class name

`$associatedclass=ACME_Device`

specifies filtering on the associated class name "ACME_Device"

6.6.2 $associatedrole (associated role filter)

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

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

```
query-parameter =/ associatedrole-query-parm

associatedrole-query-parm = "$associatedrole=" reference-name
```

Where:

- The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)
- `reference-name` is the name of the reference property referencing the associated class. Note that CIM property names do not contain reserved characters (see 6.3 and DSP0004)

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

Examples:

(not specified)

specifies no filtering on the associated role name

`$associatedrole=Device`

specifies filtering on the associated role name "Device"

6.6.3 $associationclass (association class filter)

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

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

```
query-parameter =/ associationclass-query-parm

associationclass-query-parm = "$associationclass=" class-name
```

Where:

- The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)
- `class-name` is the name of the association class (including schema prefix). Note that CIM class names do not contain reserved characters (see 6.3 and DSP0004)

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

Examples:

(not specified)

specifies no filtering on the association class name

`$associationclass=ACME_SystemDevice`

specifies filtering on the association class name "ACME_SystemDevice"

6.6.4 $class (specify class name)

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

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

```
query-parameter =/ class-query-parm

class-query-parm = "$class=" class-name
```

Where:

- The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)
- `class-name` is the name of the class (including schema prefix). Note that CIM class names do not contain reserved characters (see 6.3 and DSP0004)

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

Examples:

(not specified)

specifies no class name

```
$class=ACME_ComputerSystem
```

specifies class name "ACME_Computersystem"

6.6.5 $continueonerror (continue on errors within paged retrieval)

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

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

```
query-parameter =/ continueonerror-query-parm

continueonerror-query-parm = "$continueonerror=" ( "true" / "false" )
```

Where:

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

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

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

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

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

Examples:

(not specified)
`$continueonerror=false`

The server closes paged retrieval sequences in case of errors

`$continueonerror=true`

The server continues paged retrieval sequences in case of errors

6.6.6 $filter (filter instances in result)

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

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

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

```
query-parameter =/ filter-query-parm

filter-query-parm = "$filter=" [ filter-query ]
```

Where:

- The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded).
- `filter-query` is a filter query string that shall conform to the format of the filter language specified with the `$filterlanguage` parameter (or its default if not specified); if it evaluates to true for an instance then the instance is included, otherwise, it is not included.

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

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

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

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

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

Examples (using FQL as a filter language):

(not specified)

no additional restrictive instance filtering takes place

`$filter=`

includes no instances

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

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

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

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

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

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

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

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

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

6.6.7 $filterlanguage (specify filter language)

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

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

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

```
query-parameter =/ filterlanguage-query-parm

filterlanguage-query-parm = "$filterlanguage=" filter-language
```

Where:

- The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded).
- `filter-language` specifies the filter query language, using an identifier defined by the filter language specification. The filter language is treated case-insensitively.

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

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

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

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

Examples:

(not specified)

FQL is used by default

`$filterlanguage=DMTF%3AFQL`

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

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

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

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

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

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

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

max-members = nonNegativeDecimalInteger
```

Where:

- The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)
- `max-members` specifies the maximum number of collection members.

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

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

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

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

Examples:

(not specified)

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

`$max=0`

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

`$max=10`

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

6.6.9 $pagingtimeout (specify inactivity timeout for paged retrieval)

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

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

```
query-parameter =/ pagingtimeout-query-parm

pagingtimeout-query-parm = "$pagingtimeout=" duration

duration = nonNegativeDecimalInteger
```

Where:

- The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)
- `duration` is the duration of the paging timeout in seconds. A value of 0 specifies that there is no paging timeout (that is, an infinite paging timeout)

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

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

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

Examples:

(not specified)

a paging timeout specific to the server implementation is used

`$pagingtimeout=0`

no paging timeout is used (infinite paging timeout)

`$pagingtimeout=30`

a paging timeout of 30 seconds is used

6.6.10 $properties (subset properties in result)

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

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

```
query-parameter =/ properties-query-parm

properties-query-parm = "$properties=" [ property-list ]

property-list = property-name *( "," property-name )
```

Where:

- The reserved characters "$", "=" and "," in the literals of these ABNF rules shall be considered delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded).
- `property-name` is the name of a property in the instances. Note that CIM property names do not contain reserved characters (see 6.3 and DSP0004).

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

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

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

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

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

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

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

Examples:

(not specified)

no properties are excluded

`$properties=`

no properties are included

`$properties=Name,Type`

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

6.6.11 $qualifiers (include qualifiers in returned classes)

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

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

```
query-parameter =/ qualifiers-query-parm

qualifiers-query-parm = "$qualifiers=" ( "true" / "false" )
```

Where:

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

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

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

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

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

Examples:

```
(not specified)
$qualifiers=false
```

No qualifiers are included in any returned classes.

```
$qualifiers=true
```

Qualifiers are included in any returned classes.

6.6.12 $sourcerole (source role filter)

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

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

```
query-parameter =/ sourcerole-query-parm

sourcerole-query-parm = "$sourcerole=" reference-name
```

Where:

- The reserved characters "$" and "=" in the literals of these ABNF rules shall be considered delimiters for purposes of percent-encoding (see 6.3, that is, they shall not be percent-encoded)
- `reference-name` is the name of the reference property referencing the source class. Note that CIM property names do not contain reserved characters (see 6.3 and DSP0004)

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

Examples:

(not specified)

specifies no filtering on the source role name

`$sourcerole=System`

specifies filtering on the source role name "System"

6.6.13 $subclasses (include subclasses in class enumeration result)

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

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

```
query-parameter =/ subclasses-query-parm

subclasses-query-parm = "$subclasses=" ( "true" / "false" ) ]
```

Where:

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

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

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

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

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

Examples:

```
(not specified)
$subclasses=false
```

No subclasses are included into the class collection.

```
$subclasses=true
```

Subclasses are included into the class collection.

7 Resources, operations and payload elements

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

7.1 Overview

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

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

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

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

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

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

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

Table 4 – CIM-RS payload elements

Payload Element	Meaning	Description
Instance	Representation of an instance resource; that is, a modeled object in the managed environment	See 7.5.2
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

7.2 Description conventions

7.2.1 Data types used in payload element definitions

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

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

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

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. Condition: There are such qualifiers.
array	Boolean	Conditional	specifies whether the element is an array. Condition: The element is an array. Default if not specified: False.
arraysize	Integer, or None	Conditional	specifies the size of the fixed-size array. Condition: The array is an array. A value of NULL indicates that the array is variable-sized. 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. Condition: The CIM data type requires a class name to be specified, see Table 6. Default if not specified: Not applicable.
defaultvalue	Value	Conditional	default value for the property. Condition: The represented element is a property and the property has a non-Null default value. 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

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

7.2.2 Data type names

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

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

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

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.

7.2.4 Requirement levels used in operation definitions

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

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).

7.2.5 Description format for operations

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

Purpose:	A brief description of the purpose of the operation.
HTTP method:	The name of the HTTP method used to perform the operation (for example, GET, PUT, POST, DELETE).
Target resource:	The type of resource that is identified as the target of the HTTP method, by means of the Request-URI field (see RFC2616) and Host header field.
Query parameters:	The names of any query parameters that may be specified in the resource identifier. Other query parameters shall not be specified by the requester. If other query parameters are specified by the requester, they shall be ignored by the responder, in order to provide for future extensibility.
Request headers:	The names of any header fields that may be specified in the request message. Other request headers shall not be specified by the requester. If other query request headers are specified by the requester, they shall be ignored by the responder, in order to provide for future extensibility.
Request payload:	The name of the payload element that shall be used in the entity body of the request message. "None" means the entity body shall be empty.
Response headers:	The names of any header fields that may be specified in the response message, separately for the success and failure cases. Other response headers shall not be specified by the responder. If other query request headers are specified by the responder, they shall be ignored by the requester, in order to provide for future extensibility.
Response payload:	The name of the payload element that shall be used in the entity body of the response message, separately for the success and failure cases. "None" means the entity body shall be empty.
Requirement:	The requirement level to implement the operation, as defined in 7.2.4.
Description:	A normative definition of the behavior of the operation, in addition to the normative definitions stated in this subclause. Normative requirements in this subclause are sometimes directed to the provider of the operation, and sometimes to its consumer.
Example HTTP conversation:	An example HTTP request and HTTP response. The examples are informative and use the CIM-RS payload representation in JSON as defined in DSP0211. They do not show all cases of using query parameters or all cases of including or not including type information (a concept supported by DSP0211) In case of differences between these examples and DSP0211, the latter is authoritative.

7.3 Common behaviors for all operations

7.3.1 Content negotiation

In order to determine the type of CIM-RS payload representation to be used, WBEM clients, servers, and listeners shall support server-driven content negotiation as defined in RFC2616, based on the Accept request-header (defined in RFC2616 and in 8.4.1), and the Content-Type response header field (defined in RFC2616 and in 8.4.2).

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

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

7.3.2 Caching of responses

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

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

7.3.3 Success and failure

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

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

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

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

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

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

7.3.4 Errors

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

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

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

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

7.3.5 ErrorResponse payload element

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

An ErrorResponse payload element shall have the following attributes:

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.

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

Response (if type information is included):

```
HTTP/1.1 404 Not Found
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "errorresponse",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
ys11",
  "httpmethod": "GET",
  "statuscode": 6,
  "statusdescription": "WIPG0213: CIM instance ACME_VirtualSystem.InstanceID=\"node
47:sys11\" does not exist in CIM namespace root/cimv2.",
  "errors": [
    {
      "kind": "instance",
      // self is omitted for embedded instances
      // namespace is omitted for embedded instances
      "classname": "CIM_Error",
      "properties": {
        "ErrorType": {
          "type": "uint16",
          "value": 4},
        "ErrorSource": {
          "type": "string",
          "value": "root/cimv2:ACME_VirtualSystem.InstanceID=\"node47:sys11\""},
        "ErrorSourceFormat": {
          "type": "uint16",
          "value": 2},
        "Message": {
          "type": "string",
          "value": "WIPG0213: CIM instance ACME_VirtualSystem.InstanceID=\"node47:s
ys11\" does not exist in CIM namespace root/cimv2."},
        "MessageArguments": {
          "type": "string",
          "array": true,
          // arraysize is omitted
          "value": [
            "ACME_VirtualSystem.InstanceID=\"node47:sys11\"",
            "root/cimv2",
            "GetInstance",
            null,
            "root/cimv2:ACME_VirtualSystem.InstanceID=\"node47:sys11\""
        ]},
        "MessageID": ": {
          "type": "string",
          "value": "WIPG0213"},
        "OwningEntity": {
          "type": "string",
          "value": "DMTF"}
      }
    }
  ]
}
```

7.3.6 Consistency model

The operations of the CIM-RS protocol shall conform to the consistency model defined in DSP0223.

7.3.7 Paging of instance collections

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

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

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

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

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

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

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

Table 10 lists other operations related to paged instance collections:

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

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

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

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

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

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

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

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

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

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

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

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

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

7.4 Optional features of the CIM-RS protocol

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

7.4.1 "Continue on error" feature

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

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

7.5 Instance resource

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

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

7.5.1 Resource identifier

Instance resources shall have a resource identifier whose path component (that is, the `path-absolute` 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
```

Where:

- `nsname` is the namespace name, in its original lexical case, percent-encoded as defined in 6.3. The reserved character "/" in namespace names shall be considered data for purposes of percent-encoding (that is, it shall be percent-encoded); otherwise, namespace names do not contain reserved characters.
- `classname` is the class name, in its original lexical case, percent-encoded as defined in 6.3. Note that CIM class names do not contain reserved characters (see 6.3 and DSP0004).

- `keyname` is the key property name, in its original lexical case, percent-encoded as defined in 6.3. Note that CIM property names do not contain reserved characters (see 6.3 and DSP0004).
- `keyvalue` is the key property value. The character sequence used for this resource identifier component shall be the string representation of the CIM typed value as defined in DSP0004, with any reserved characters considered to be data (see 6.3, that is, they shall be percent-encoded).

Examples:

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

7.5.2 Instance payload element

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

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

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.

7.5.3 MethodRequest payload element

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

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

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.

7.5.4 MethodResponse payload element

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

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

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.

7.5.5 GET (retrieve an instance)

Purpose:	Retrieve an instance
HTTP method:	GET
Target resource:	Instance (see 7.5.1)
Query parameters:	`$properties`
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None

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

Response payload (success): Instance (see 7.5.2)

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

Response payload (failure): ErrorResponse (see 7.3.5)

Requirement: Class-specific

Description:

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

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

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

- 200 "OK": The entity body shall contain an Instance payload element representing the specified instance (see 7.5.2).
- 304 "Not Modified": The validators matched on a conditional request; the entity body shall be empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request. For details on conditional requests, see subclause 9.3 in RFC2616.

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

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

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

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

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

Response (if type information is included):

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "instance",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
ys11",
  "namespace": "root/cimv2",
  "classname": "ACME_VirtualSystem",
  "properties": {
    "InstanceID": {
      "type": "string",
      "value": "node47:sys11" },
    "ElementName": {
      "type": "string",
      "value": "Virtual system 11 on node 07" },
    "Caption": {
      "type": "string",
      "value": "Virtual system 11 on node 07" },
    . . .
  }
}
```

7.5.6 PUT (update an instance)

Purpose:	Update an instance (partially or fully)
HTTP method:	PUT
Target resource:	Instance (see 7.5.1)
Query parameters:	`$properties`
Request headers:	Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
Request payload:	Instance (see 7.5.2)
Response headers (success):	Date, X-CIMRS-Version

Response payload (success): None

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

Response payload (failure): ErrorResponse (see 7.3.5)

Requirement: Class-specific

Description:

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

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

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

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

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

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

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

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

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

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

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

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

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

```
PUT /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11 HT
TP/1.1
Host: server.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.0

{
  "kind": "instance",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
ys11",
  "namespace": "root/cimv2",
  "classname": "ACME_VirtualSystem",
  "properties": {
    // InstanceID is not included because it is not updateable
    "ElementName": {
      "type": "string",
      "value": "Tom's system" },
    "Caption": {
      "type": "string",
      "value": "Tom's system (sys 11 on node 47)" },
    . . .    // all other updateable properties
  }
}
```

Response:

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

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

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

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

```
PUT /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11?$p
roperties=ElementName,Caption HTTP/1.1
Host: server.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.0

{
  "kind": "instance",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
ys11?$properties=ElementName,Caption",
  "namespace": "root/cimv2",
  "classname": "ACME_VirtualSystem",
  "properties": {
    "ElementName": {
      "type": "string",
      "value": "Tom's system" },
    "Caption": {
      "type": "string",
      "value": "Tom's system (sys 11 on node 47)" }
  }
}
```

Response:

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

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

7.5.7 DELETE (delete an instance)

Purpose:	Delete an instance
HTTP method:	DELETE
Target resource:	Instance (see 7.5.1)
Query parameters:	None
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, X-CIMRS-Version
Response payload (success):	None
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Class-specific

Description:

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

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

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

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

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

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

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

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

Response:

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

7.5.8 POST (invoke a method on an instance)

Purpose:	Invoke a method on an instance
HTTP method:	POST
Target resource:	Instance (see 7.5.1)
Query parameters:	None
Request headers:	Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
Request payload:	MethodRequest (see 7.5.3)
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (success):	MethodResponse (see 7.5.4)
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Class-specific

Description:

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

The method may be static or non-static.

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

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

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

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

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

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

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

```
POST /root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3Asys11 H
TTP/1.1
Host: server.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.0

{
  "kind": " methodrequest",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
ys11",
  "methodname": "RequestStateChange",
  "parameters": {
    "RequestedState": {
      "type": "uint16",
      "value": 2 },
    "TimeoutPeriod": {
      "type": "datetime",
      "value": None }
  }
}
```

Response (if type information is included):

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "methodresponse",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
ys11",
  "methodname": "RequestStateChange",
  "returnvalue": {
    "type": "uint32",
    "value": 0 },
  "parameters": {
    "Job": {
      "type": "reference",
      "classname": "ACME_Job",
      "value": None },
  }
}
```

7.6 Instance collection resource

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

7.6.1 Resource identifier

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

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

Where:

- `nsname` is the namespace name, in its original lexical case, percent-encoded as defined in 6.3. The reserved character "/" in namespace names shall be considered data for purposes of percent-encoding (that is, it shall be percent-encoded); otherwise, namespace names do not contain reserved characters.
- `classname` is the class name, in its original lexical case, percent-encoded as defined in 6.3. Note that CIM class names do not contain reserved characters (see 6.3 and DSP0004).

Examples:

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

7.6.2 InstanceCollection payload element

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

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

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

7.6.3 POST (create an instance)

Purpose:	Create an instance
HTTP method:	POST
Target resource:	Instance collection (see 7.6.1)

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

Description:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
POST /root%2Fcimv2/classes/ACME_VirtualSystem/instances HTTP/1.1
Host: server.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.0

{
  "kind": "instance",
  // self is omitted in creation
  "namespace": "root/cimv2",
  "classname": "ACME_VirtualSystem",
  "properties": {
    "ElementName": {
      "type": "string",
      "value": "Tom's system" },
    // Other initial property values
  }
}
```

Response:

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

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

7.6.4 GET (enumerate instances of a class)

Purpose:	Enumerate instances of a class
HTTP method:	GET
Target resource:	Instance collection (see 7.6.1)
Query parameters:	`$properties`, `$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
Response payload (success):	InstanceCollection (see 7.6.2), may be paged
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Class-specific

Description:

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

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

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

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

- 200 "OK": The entity body shall contain an InstanceCollection payload element representing the returned instances (see 7.6.2). The collection may be empty.
- 304 "Not Modified": The validators matched on a conditional request; the entity body shall be empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request. For details on conditional requests, see subclause 9.3 in RFC2616.

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

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

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

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

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

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

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "instancecollection",
  "self": "/root%2Fcimv2/classes/ACME_ComputerSystem/instances",
  "instances": [
    {
      "kind": "instance",
      "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47
%3Asys11",
      "namespace": "root/cimv2",
      "classname": "ACME_ComputerSystem",
      "properties": {
        "InstanceID": {
          "type": "string",
          "value": "node47:sys11" },
        "ElementName": {
          "type": "string",
          "value": "Tom's system" },
        // Other property values of this instance
      }
    },
    // Other instances of this class
  ]
}
```

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

7.7 Instance associator collection resource

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

7.7.1 Resource identifier

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

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

Where:

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

7.7.2 GET (retrieve associated instances)

Purpose:	Retrieve associated instances
HTTP method:	GET
Target resource:	Instance associator collection (see 7.7.1)
Query parameters:	`$associationclass`, `$sourcerole`, `$associatedclass`, `$associatedrole`, `$properties`, `$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
Response payload (success):	InstanceCollection (see 7.6.2), may be paged
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Class-specific

Description:

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

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

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

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

- 200 "OK": The entity body shall contain an InstanceCollection payload element representing the returned instances (see 7.6.2). The collection may be empty.
- 304 "Not Modified": The validators matched on a conditional request; the entity body shall be empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request. For details on conditional requests, see subclause 9.3 in RFC2616.

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

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

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

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

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

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

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "instancecollection",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
ys11/associators",
  "instances": [
    {
      "kind": "instance",
      "self": "/root%2Fcimv2/classes/ACME_NetworkInterface/instances/InstanceID=nod
e47%3Asys11%3Aeth0",
      "namespace": "root/cimv2",
      "classname": "ACME_NetworkInterface",
      "properties": {
        "InstanceID": {
          "type": "string",
          "value": "eth0" },
        "IPAddress": {
          "type": "string",
          "value": "10.11.12.13" },
        . . .  // Other properties of this instance
      }
    },
    . . .  // Other associated instances
  ]
}
```

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

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

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

Response payload (failure): ErrorResponse (see 7.3.5)

Requirement: Class-specific

Description:

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

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

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

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

- 200 "OK": The entity body shall contain an InstanceCollection payload element representing the returned instances (see 7.6.2). The collection may be empty.
- 304 "Not Modified": The validators matched on a conditional request; the entity body shall be empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request. For details on conditional requests, see subclause 9.3 in RFC2616.

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

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

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

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

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

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

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "instancecollection",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
ys11/references",
  "instances": [
    {
      "kind": "instance",
      "self": "/root%2Fcimv2/ACME_SystemNetworkDevice/System=. . .,Device=. . .",
      "namespace": "root/cimv2",
      "classname": "ACME_SystemNetworkDevice",
      "properties": {
        "System": {
          "type": "reference",
          "classname": "ACME_VirtualSystem",
          "value": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=n
ode47%3Asys11" },
        "Device": {
          "type": "reference",
          "classname": "ACME_NetworkInterface",
          "value": "/root%2Fcimv2/classes/ACME_NetworkInterface/instances/InstanceI
D=node47%3Asys11%3Aeth0" },
          . . .  // Other property values of this association instance
        }
      },
      . . .  // Other referencing association instances
  ]
}
```

7.9 Instance collection page resource

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

7.9.1 Resource identifier

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

7.9.2 GET (retrieve instance collection page)

Purpose:	Retrieve instance collection page
HTTP method:	GET
Target resource:	Instance collection page (see 7.9.1)
Query parameters:	`$max`
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (success):	InstanceCollection (see 7.6.2)
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Class-specific

Description:

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

For details on paged retrieval, see 7.3.7.

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

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

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

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.

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

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

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

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

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

Response (if type information is included):

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "instancecollection",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node47%3As
ys11/references/page/123456",
  "instances": [
    {
      "kind": "instance",
      "self": "/root%2Fcimv2/ACME_SystemNetworkDevice/System=. . .,Device=. . .",
      "namespace": "root/cimv2",
      "classname": "ACME_SystemNetworkDevice",
      "properties": {
        "System": {
          "type": "reference",
          "classname": "ACME_VirtualSystem",
          "value": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=n
ode47%3Asys11" },
        "Device": {
          "type": "reference",
          "classname": "ACME_NetworkInterface",
          "value": "/root%2Fcimv2/classes/ACME_NetworkInterface/instances/InstanceI
D=node47%3Asys11%3Aeth0" },
        . . .  // Other property values of this instance
      }
    },
    . . .  // Other instances in this page
  ]
}
```

7.9.3 DELETE (close paged instance collection)

Purpose:	Close paged instance collection
HTTP method:	DELETE
Target resource:	Instance collection page (see 7.9.1)
Query parameters:	None
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, X-CIMRS-Version
Response payload (success):	None
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Class-specific

Description:

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

For details on paged retrieval, see 7.3.7.

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

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

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

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

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

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

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

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

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

Response:

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

7.10 Class resource

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

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

7.10.1 Resource identifier

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

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

Where:

- `nsname` is the namespace name, in its original lexical case, percent-encoded as defined in 6.3. The reserved character "/" in namespace names shall be considered data for purposes of percent-encoding (that is, it shall be percent-encoded); otherwise, namespace names do not contain reserved characters.
- `classname` is the class name, in its original lexical case, percent-encoded as defined in 6.3. Note that CIM class names do not contain reserved characters (see 6.3 and DSP0004).

Examples:

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

7.10.2 Class payload element

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

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

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.

7.10.3 GET (retrieve a class)

Purpose:	Retrieve a class
HTTP method:	GET
Target resource:	Class (see 7.10.1)
Query parameters:	`$qualifiers`
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (success):	Class (see 7.10.2)
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Optional

Description:

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

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

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

- 200 "OK": The entity body shall contain a Class payload element representing the returned class (see 7.10.2).
- 304 "Not Modified": The validators matched on a conditional request; the entity body shall be empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request. For details on conditional requests, see subclause 9.3 in RFC2616.

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

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

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

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

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

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

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "class",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem",
  "namespace": "root/cimv2",
  "name": "ACME_VirtualSystem",
  "superclassname": "ACME_ComputerSystem",
  "qualifiers": {
    "Description": {
      "type": "string",
      "value": "A virtual system.\n . . ."
    },
    . . .  // Other qualifier values for this class
  },
  "properties": {
    "InstanceID": {
      "qualifiers" : { . . . },
      // array and arraysize are omitted
      "type": "string"
    },
    "ElementName": {
      "qualifiers" : { . . . },
      "default": "",
      // array and arraysize are omitted
      "type": "string"
    },
    . . .  // Other property definitions for this class
  },
  "methods": {
    "RequestStateChange": {
      "qualifiers" : { . . . },
      // array and arraysize are omitted
      "type": "uint32"
      "parameters": {
        "RequestedState": {
          "qualifiers" : { . . . },
          // array and arraysize are omitted
          "type": "uint16"
        },
        . . .  // Other parameters of this method
      }
    },
    . . .  // Other method definitions for this class
  }
}
```

7.10.4 PUT (update a class)

Purpose:	Update a class
HTTP method:	PUT
Target resource:	Class (see 7.10.1)
Query parameters:	None
Request headers:	Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
Request payload:	Class (see 7.10.2)
Response headers (success):	Date, X-CIMRS-Version
Response payload (success):	None
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Optional

Description:

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

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

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

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

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

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

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

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

```
PUT /root%2Fcimv2/classes/ACME_VirtualSystem HTTP/1.1
Host: server.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.0

{
  "kind": "class",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem",
  "namespace": "root/cimv2",
  "name": "ACME_VirtualSystem",
  "superclassname": "ACME_ComputerSystem",
  "qualifiers": { . . . },
  "properties": { . . . },
  "methods": { . . . }
}
```

Response:

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

7.10.5 DELETE (delete a class)

Purpose:	Delete a class
HTTP method:	DELETE
Target resource:	Class (see 7.10.1)
Query parameters:	None
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, X-CIMRS-Version
Response payload (success):	None
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Optional

Description:

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

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

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

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

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

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

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

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

Response:

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

7.10.6 POST (invoke a method on a class)

Purpose:	Invoke a method on a class
HTTP method:	POST
Target resource:	Class (see 7.10.1)
Query parameters:	None
Request headers:	Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
Request payload:	MethodRequest (see 7.5.3)
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (success):	MethodResponse (see 7.5.4)
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Class-specific

Description:

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

The method shall be static.

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

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

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 29 or Table 7 shall be returned.

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

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

Example HTTP conversation for invocation of static method (using JSON as defined in DSP0211):

Request (if type information is included):

```
POST /root%2Fcimv2/classes/ACME_VirtualSystem HTTP/1.1
Host: server.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.0

{
  "kind": "methodrequest",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem",
  "method": "CreateVirtualSystem",
  "parameters": {
    "Template": {
      "type": "string",
      "value": "small" }
  }
}
```

Response (if type information is included):

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "methodresponse",
  "self": "/root%2Fcimv2/classes/ACME_VirtualSystem",
  "method": "CreateVirtualSystem",
  "returnvalue": {
    "type": "uint32",
    "value": 0 },
  "parameters": {
    "System": {
      "type": "reference",
      "classname": "ACME_VirtualSystem",
      "value": "/root%2Fcimv2/classes/ACME_VirtualSystem/instances/InstanceID=node4
7%3Asys12" }
  }
}
```

7.11 Class collection resource

A class collection resource represents a list of class resources.

7.11.1 Resource identifier

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

```
class-coll-path-absolute = "/" nsname "/classes"
```

Where:

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

Examples:

```
/root%2Fcimv2/classes
```

7.11.2 ClassCollection payload element

A ClassCollection payload element is the representation of a class collection resource in the protocol.

Unless otherwise constrained, a ClassCollection payload element shall have the attributes defined in Table 30.

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.

7.11.3 POST (create a class)

Purpose: Create a class

HTTP method: POST

Target resource: Class collection (see 7.11.1)

Query parameters: None

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

Request payload: Class (see 7.10.2), without the "self" attribute

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

Response payload (success): None

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

Response payload (failure): ErrorResponse (see 7.3.5)

Requirement: Optional

Description:

The HTTP POST method on a class collection resource creates the specified class in the namespace of that class collection.

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

The qualifiers, properties and methods for the new class are defined in a class representation in the payload.

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

The "namespace" attribute in the request payload element is optional. If specified, it shall be consistent with the target resource identifier.

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

- 201 "Created": The entity body shall be empty and the "Location" header field shall be set to the resource identifier of the newly created class.

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 31 or Table 7 shall be returned.

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

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

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

```
POST /root%2Fcimv2/classes HTTP/1.1
Host: server.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.0

{
  "kind": "class",
  // self is omitted in creation
  "namespace": "root/cimv2",
  "name": "ACME_VirtualSystem",
  "superclassname": "ACME_ComputerSystem",
  "qualifiers": { . . . },
  "properties": { . . . },
  "methods": { . . . }
}
```

Response:

```
HTTP/1.1 201 Created
Date: Thu, 30 Oct 2014 15:03:00 GMT
Location: //server.acme.com:5988/root%2Fcimv2/classes/ACME_VirtualSystem
X-CIMRS-Version: 2.0.1
```

7.11.4 GET (enumerate classes)

Purpose:	Enumerate classes
HTTP method:	GET
Target resource:	Class collection (see 7.11.1)
Query parameters:	`$class`, `$subclasses`, `$qualifiers`
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (success):	ClassCollection (see 7.11.2)
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Optional

Description:

The HTTP GET method on a class collection resource enumerates top-level classes in a namespace or subclasses of a specified class.

The set of classes included in the result depends on both the `$class` and `$subclasses` query parameters, as follows:

- An intermediate set of classes is determined, as follows: If query parameter `$class` (see 6.6.4) is specified, the direct subclasses of the specified class are in the intermediate set. Otherwise, the top-level classes in the namespace identified of the target resource identifier are in the intermediate set.
- The value of the `$subclasses` query parameter (6.6.13) governs whether the intermediate set of classes becomes the result set (if false), or (if true) is amended by the direct and indirect subclasses of each class in the intermediate set.

Qualifier values shall be returned for each returned class resource if the `$qualifiers` parameter (6.6.11) evaluates to true.

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

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

- 200 "OK": The entity body shall contain a ClassCollection payload element representing the returned classes (see 7.11.2). The collection may be empty.
- 304 "Not Modified": The validators matched on a conditional request; the entity body shall be empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request. For details on conditional requests, see subclause 9.3 in RFC2616.

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 32 or Table 7 shall be returned.

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

Example HTTP conversation for enumerating the direct subclasses of a class (using JSON as defined in DSP0211):

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

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

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

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "classcollection",
  "self": "/root%2Fcimv2/classes?$class=ACME_ComputerSystem",
  "classes": [
    {
      "kind": "class",
      "self": "/root%2Fcimv2/classes/ACME_VirtualSystem",
      "namespace": "root/cimv2",
      "name": "ACME_VirtualSystem",
      "superclassname": "ACME_ComputerSystem",
      "qualifiers": { . . . },
      "properties": { . . . },
      "methods": { . . . }
    },
    . . .  // Other direct subclasses of ACME_ComputerSystem
  ]
}
```

7.12 Class associator collection resource

A class associator collection resource represents the classes associated to a source class.

7.12.1 Resource identifier

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

```
class-associator-coll-path-absolute = class-path-absolute "/associators"
```

Where:

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

7.12.2 GET (retrieve associated classes)

Purpose:	Retrieve associated classes
HTTP method:	GET
Target resource:	Class associator collection (see 7.12.1)
Query parameters:	`$associationclass`, `$sourcerole`, `$associatedclass`, `$associatedrole`, `$qualifiers`
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (success):	ClassCollection (see 7.11.2), may be paged
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Optional

Description:

The HTTP GET method on a class associator collection resource analyzes the class structure starting on a source class and returns a class collection with representations of the classes associated with the source class.

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

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

- 200 "OK": The entity body shall contain a ClassCollection payload element representing the returned classes (see 7.11.2). The collection may be empty.
- 304 "Not Modified": The validators matched on a conditional request; the entity body shall be empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request. For details on conditional requests, see subclause 9.3 in RFC2616.

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 33 or Table 7 shall be returned.

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

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

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

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

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

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "classcollection",
  "self": "/root%2Fcimv2/classes/ACME_ComputerSystem/associators",
  "classes": [
    {
      "kind": "class",
      "self": "/root%2Fcimv2/classes/ACME_NetworkInterface",
      "namespace": "root/cimv2",
      "name": "ACME_NetworkInterface",
      "superclassname": "ACME_Device",
      "qualifiers": { . . . },
      "properties": { . . . },
      "methods": { . . . }
    },
    . . .  // Other associated classes
  ]
}
```

7.13 Class reference collection resource

A class reference collection resource represents the association classes referencing a source class.

7.13.1 Resource identifier

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

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

Where:

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

7.13.2 GET (retrieve referencing classes)

Purpose:	Retrieve referencing classes
HTTP method:	GET
Target resource:	Class reference collection (see 7.13.1)
Query parameters:	`$associationclass`, `$sourcerole`, `$qualifiers`
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (success):	ClassCollection (see 7.11.2), may be paged
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Optional

Description:

The HTTP GET method on a class reference collection resource analyzes the class structure starting on a source class and returns a class collection with representations of the association classes referencing the source class.

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

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

- 200 "OK": The entity body shall contain a ClassCollection payload element representing the returned classes (see 7.11.2). The collection may be empty.
- 304 "Not Modified": The validators matched on a conditional request; the entity body shall be empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request. For details on conditional requests, see subclause 9.3 in RFC2616.

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 34 or Table 7 shall be returned.

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

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

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

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

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

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
```

```
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "classcollection",
  "self": "/root%2Fcimv2/classes/ACME_ComputerSystem/references",
  "classes": [
    {
      "kind": "class",
      "self": "/root%2Fcimv2/classes/ACME_SystemDevice",
      "namespace": "root/cimv2",
      "name": "ACME_SystemDevice",
      // no superclass
      "qualifiers": { . . . },
      "properties": { . . . },
      // no methods
    },
    . . .  // Other referencing classes
  ]
}
```

7.14 Qualifier type resource

A qualifier type resource represents a CIM qualifier type (that is, the declaration of a qualifier).

7.14.1 Resource identifier

Qualifier type resources shall have a resource identifier whose path component (that is, the `path-absolute` ABNF rule in 6.1) matches ABNF rule `qualifiertype-path-absolute`:

```
qualifiertype-path-absolute = "/" nsname "/qualifiertypes/" qualifiername
```

Where:

- `nsname` is the namespace name, in its original lexical case, percent-encoded as defined in 6.3. The reserved character "/" in namespace names shall be considered data for purposes of percent-encoding (that is, it shall be percent-encoded); otherwise, namespace names do not contain reserved characters.
- `qualifiername` is the qualifier name, percent-encoded as defined in 6.3. Note that CIM qualifier names do not contain reserved characters (see 6.3 and DSP0004).

Examples:

```
/root%2Fcimv2/qualifiertypes/Abstract
```

7.14.2 QualifierType payload element

A QualifierType payload element is the representation of a qualifier type in the protocol.

Unless otherwise constrained, a QualifierType payload element shall have the attributes defined in Table 35.

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.

7.14.3 GET (retrieve a qualifier type)

Purpose:	Retrieve a qualifier type
HTTP method:	GET
Target resource:	Qualifier type (see 7.14.1)
Query parameters:	None
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version

Response payload (success): QualifierType (see 7.14.2)

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

Response payload (failure): ErrorResponse (see 7.3.5)

Requirement: Optional

Description:

The HTTP GET method on a qualifier type resource retrieves a representation of the specified qualifier type.

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

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

- 200 "OK": The entity body shall contain a QualifierType payload element representing the returned qualifier type (see 7.14.2).
- 304 "Not Modified": The validators matched on a conditional request; the entity body shall be empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request. For details on conditional requests, see subclause 9.3 in RFC2616.

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 36 or Table 7 shall be returned.

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

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

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

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

Response (if type information is included. Note that the inclusion of type information does not influence the representation of qualifier types. For details, see DSP0211):

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.1;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "qualifiertype",
  "self": "/root%2Fcimv2/qualifiertypes/Abstract",
  "namespace": "root/cimv2",
  "name": "Abstract",
  "type": "boolean",
  "scopes": ["class", "association", "indication"],
  "propagation": false,
  // override is omitted
  // translatable is omitted
}
```

7.14.4 PUT (update a qualifier type)

Purpose:	Update a qualifier type
HTTP method:	PUT
Target resource:	Qualifier type (see 7.14.1)
Query parameters:	None
Request headers:	Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
Request payload:	QualifierType (see 7.14.2)
Response headers (success):	Date, X-CIMRS-Version
Response payload (success):	None
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Optional

Description:

The HTTP PUT method on a qualifier type resource updates the entire resource with the specified qualifier type representation.

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

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

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

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 37 or Table 7 shall be returned.

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

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

Request (if type information is included in the request and accepted to be included in an error response. Note that the inclusion of type information does not influence the representation of qualifier types. For details, see DSP0211):

```
PUT /root%2Fcimv2/Abstract HTTP/1.1
Host: server.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.0

{
  "kind": "qualifiertype",
  "self": "/root%2Fcimv2/qualifiertypes/Abstract",
  "namespace": "root/cimv2",
  "name": "Abstract",
  "type": "boolean",
  "scopes": ["class", "association", "indication"],
  "propagation": false,
  // override is omitted
  // translatable is omitted
}
```

Response:

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

7.14.5 DELETE (delete a qualifier type)

Purpose:	Delete a qualifier type
HTTP method:	DELETE
Target resource:	Qualifier type (see 7.14.1)
Query parameters:	None
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, X-CIMRS-Version
Response payload (success):	None
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Optional

Description:

The HTTP DELETE method on a qualifier type resource deletes the qualifier type in its namespace.

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

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

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 38 or Table 7 shall be returned.

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

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

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

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

Response:

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

7.15 Qualifier type collection resource

A qualifier type collection resource represents a list of qualifier types.

7.15.1 Resource identifier

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

```
qualifiertype-coll-path-absolute = "/" nsname "/qualifiertypes"
```

Where:

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

Examples:

```
/root%2Fcimv2/qualifiertypes
```

7.15.2 QualifierTypeCollection payload element

A QualifierTypeCollection payload element is the representation of a qualifier type collection resource in the protocol.

Unless otherwise constrained, a QualifierTypeCollection payload element shall have the attributes defined in Table 39.

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.

7.15.3 POST (create a qualifier type)

Purpose: Create a qualifier type

HTTP method: POST

Target resource: Qualifier type collection (see 7.15.1)

Query parameters: None

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

Request payload: QualifierType (see 7.14.2), without the "self" attribute

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

Response payload (success): None

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

Response payload (failure): ErrorResponse (see 7.3.5)

Requirement: Optional

Description:

The HTTP POST method on a qualifier type collection resource creates the specified qualifier type in the namespace of that collection.

On return, the Location header specifies the resource identifier of the newly created qualifier type.

The attributes for the new qualifier type are defined in a qualifier type representation in the payload.

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

The "namespace" attribute in the request payload element is optional. If specified, it shall be consistent with the target resource identifier.

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

- 201 "Created": The entity body shall be empty and the "Location" header field shall be set to the resource identifier of the newly created qualifier type.

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 40 or Table 7 shall be returned.

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

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

Request (if type information is included in the request and accepted to be included in an error response. Note that the inclusion of type information does not influence the representation of qualifier types. For details, see DSP0211):

```
POST /root%2Fcimv2/qualifiertypes HTTP/1.1
Host: server.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.0

{
  "kind": "qualifiertype",
  // self is omitted in creation
  "namespace": "root/cimv2",
  "name": "Abstract",
  "type": "boolean",
  "scopes": ["class", "association", "indication"],
  "propagation": false,
  // override is omitted
  // translatable is omitted
}
```

Response:

```
HTTP/1.1 201 Created
Date: Thu, 30 Oct 2014 15:03:00 GMT
Location: //server.acme.com:5988/root%2Fcimv2/qualifiertypes/Abstract
X-CIMRS-Version: 2.0.1
```

7.15.4 GET (enumerate qualifier types)

Purpose:	Enumerate qualifier types
HTTP method:	GET
Target resource:	Qualifier type collection (see 7.15.1)
Query parameters:	None
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (success):	QualifierTypeCollection (see 7.15.2)
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Optional

Description:

The HTTP GET method on a qualifier type collection resource enumerates the qualifier types in the namespace of that collection.

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

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

- 200 "OK": The entity body shall contain a QualifierTypeCollection payload element representing the returned qualifier type (see 7.15.2). The collection may be empty.
- 304 "Not Modified": The validators matched on a conditional request; the entity body shall be empty. This status code can only occur if the server supports conditional requests and the client has requested a conditional request. For details on conditional requests, see subclause 9.3 in RFC2616.

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 41 or Table 7 shall be returned.

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

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

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

```
GET /root%2Fcimv2/qualifiertypes HTTP/1.1
Host: server.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
X-CIMRS-Version: 2.0.1
```

Response (if type information is included. Note that the inclusion of type information does not influence the representation of qualifier types. For details, see DSP0211):

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2014 15:03:00 GMT
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.0

{
  "kind": "qualifiertypecollection",
  "self": "/root%2Fcimv2/qualifiertypes",
  "qualifiertypes": [
    {
      "kind": "qualifiertype",
      "self": "/root%2Fcimv2/qualifiertypes/Abstract",
      "namespace": "root/cimv2",
      "name": "Abstract",
      "type": "boolean",
      "scopes": ["class", "association", "indication"],
      "propagation": false,
      // override is omitted
      // translatable is omitted
    },
    . . .  // Other qualifier types in this namespace
  ]
}
```

7.16 Listener indication delivery resource

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

7.16.1 Resource identifier

Listener indication delivery resources shall have a resource identifier whose path component (that is, the `path-absolute` ABNF rule in 6.1) matches ABNF rule `listener-indications-path-absolute`:

```
listener-indications-path-absolute = "/destinations/" destname "/indications"
```

Where:

- `destname` is the name of the listener destination, percent-encoded as defined in 6.3

Examples:

```
/destinations/srv8%3Adest1/indications
```

7.16.2 IndicationDeliveryRequest payload element

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

Unless otherwise constrained, an IndicationDeliveryRequest payload element shall have the attributes defined in Table 42.

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

7.16.3 POST (deliver an indication)

Purpose:	Deliver an indication
HTTP method:	POST
Target resource:	Listener indication delivery (see 7.16.1)
Query parameters:	None
Request headers:	Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
Request payload:	IndicationDeliveryRequest (see 7.16.2)
Response headers (success):	Date, X-CIMRS-Version
Response payload (success):	None
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.5)
Requirement:	Mandatory

Description:

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

Note that for this operation, the server decides which payload representation to use, and in case of using DSP0211, whether to include type information. In any case, the Content-Type header needs to be consistent with those decisions.

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

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

- 200 "OK": The entity body shall be empty.

On failure, the entity body shall contain an ErrorResponse payload element (see 7.3.5) and one of the HTTP status codes in Table 43 or Table 7 shall be returned.

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

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

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

```
POST /destinations/dest1/indications HTTP/1.1
Host: listener.acme.com:5988
Accept: application/vnd.dmtf.cimrs+json;version=2.0;typed=true
Content-Length: XXX
Content-Type: application/vnd.dmtf.cimrs+json;version=2.0.0;typed=true
X-CIMRS-Version: 2.0.1

{
  "kind": "indicationdeliveryrequest",
  "self": "/destinations/dest1/indications",
  "indication": {
    "kind": "instance",
    // self is omitted for embedded instances
    // namespace is omitted for embedded instances
    "classname": "ACME_AlertIndication",
    "properties": {
      "AlertType": {"type": "uint16", "value": 4},
      "PerceivedSeverity": {"type": "uint16", "value": 5},
      "ProbableCause": {"type": "uint16", "value": 42},
      "Message": {"type": "string",
                  "value": "BOND0007: Some error happened, rc=23."},
      "MessageArguments": {"type": "string", "array": True, "value": [ "23" ]},
      "MessageID": {"type": "string", "value": "BOND0007"},
      "OwningEntity": {"type": "string", "value": "ACME"}
    }
  }
}
```

Response:

```
HTTP/1.1 204 No Content
Date: Thu, 30 Oct 2014 15:03:00 GMT
X-CIMRS-Version: 2.0.0
```

7.17 CIM-RS resources to be exposed (informative)

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

7.17.1 Resources exposed by a server

For each namespace, the following resources are exposed by a server:

- Class collection resource (see 7.11)
- Qualifier type collection resource (see 7.15)

For each qualifier type in each namespace, the following resources are exposed by a server:

- Qualifier type resource (see 7.14)

For each class in each namespace, the following resources are exposed by a server:

- Class resource (see 7.10)
- Class associator collection resource (see 7.12)
- Class reference collection resource (see 7.13)
- Instance collection resource (see 7.6)

For each instance (including association instances) in each namespace, the following resources are exposed by a server:

- Instance resource (see 7.5)
- Instance associator collection resource (see 7.7)
- Instance reference collection resource (see 7.8)

For each open paged instance collection, the following resources are exposed by a server:

- Instance collection page resource (see 7.9)

In addition, resources that support query parameters have variations based upon their query parameter values.

7.17.2 Resources exposed by a listener

For each listener destination, the following resources are exposed by a listener:

- Listener indication delivery resource (see 7.16)

7.18 Other typical WBEM protocol functionality (informative)

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

7.18.1 Server discovery

WBEM servers can be discovered as described in clause 10.

7.18.2 Namespace discovery

The set of namespaces of a server can be discovered by clients using any of these approaches:

- From the Namespaces attribute of the SLP discovery data. For details, see clause 10.
- From instances of the class CIM_Namespace in the Interop namespace. See the Profile Registration Profile (DSP1033) for the concept and names of the Interop namespace. See 7.6.4 for enumerating instances of a class. Note that the use of this class for representing CIM namespaces is not covered by any DMTF standard, but is commonly implemented by WBEM servers.
- The WBEM Server Profile (DSP1092) describes how namespaces can be discovered. In short, namespaces are represented by instances of class CIM_WBEMServerNamespace in the Interop namespace. See 7.6.4 for enumerating instances of a class. This is the standards-based alternative to the previous approach, but is not yet commonly implemented by WBEM servers at the time of the publishing of version 2.0.0 of this document.

7.18.3 Registered profile discovery

The Profile Registration Profile (DSP1033) describes how to discover the management profiles to which a server advertises conformance, and from there, all further resources that are part of the functionality of a management profile. In short, the management profiles to which a server advertises conformance can be discovered by enumerating instances of class CIM_RegisteredProfile in the Interop namespace (see 7.6.4 for enumerating instances of a class).

7.18.4 Schema inspection

The schema definition (that is, class declarations and qualifier type declarations) including its meta-data in the form of qualifiers is directly accessible via the class and qualifier operations of the CIM-RS protocol (see 7.10 and following subclauses).

7.18.5 Association traversal

The CIM-RS protocol supports traversal of associations in a way consistent to generic operations (see DSP0223). For details on association traversal operations between instances, see 7.7 and 7.8. For details on association traversal operations at the class level, see 7.12 and 7.13.

7.18.6 Indication subscription

The CIM-RS protocol defines the HTTP POST method on listener indication delivery resources (see 7.16.3) for the delivery of indications (that is, event notifications). However, it does not define any specific operations for performing other indication-related functions such as subscribing for indications, retrieving and managing indication filters and filter collections, or retrieving and managing listener destinations or indication services.

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

8 HTTP usage

8.1 General requirements

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

- Version 1.1 of HTTP shall be supported as defined in RFC2616.
- Version 1.0 or earlier of HTTP shall not be supported.

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

- HTTPS shall be supported as defined in RFC2818.
- Within HTTPS, version 1.1 of HTTP shall be supported as defined in RFC2616.

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

8.2 Authentication requirements

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

8.2.1 Operating without authentication

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

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

8.2.2 HTTP basic authentication

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

WBEM clients, servers, and listeners may support HTTP basic authentication as defined in RFC2617.

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

8.2.3 HTTP digest authentication

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

WBEM clients, servers, and listeners should support HTTP digest authentication as defined in RFC2617.

8.2.4 Other authentication mechanisms

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

8.3 Message encryption requirements

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

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

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

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

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

RFC5246 describes in Appendix E "Backward Compatibility" how the secure sockets layer can be negotiated.

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

- The TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value 0x0013) shall be supported when using TLS 1.0. Note that RFC2246 defines this cipher suite to be mandatory for TLS 1.0
- The TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value 0x000A) shall be supported when using TLS 1.1. Note that RFC4346 defines this cipher suite to be mandatory for TLS 1.1
- The TLS_RSA_WITH_AES_128_CBC_SHA cipher suite (hexadecimal value 0x002F) shall be supported when using TLS 1.2. Note that RFC5246 defines this cipher suite to be mandatory for TLS 1.2
- The TLS_RSA_WITH_AES_128_CBC_SHA256 cipher suite (hexadecimal value 0x003C) should be supported when using TLS 1.2, in order to meet the transition to a security strength of 112 bits (guidance is provided in NIST Special Publication 800-57 [NIST 800-57] and NIST Special Publication 800-131A [NIST 800-131A])
- Any additional cipher suites may be supported

8.4 HTTP header fields

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

Any rules for processing header fields defined in RFC2616 apply, particularly regarding whitespace stripping, line continuation, multiple occurrences of headers, and case insensitive treatment of field names.

8.4.1 Accept

The rules for the Accept request-header field defined in RFC2616 apply. This subclause defines additional constraints on its use.

The Accept header field shall be provided on the request message of every operation. The reason is that any operation may fail and the failure response will include an ErrorResponse payload element (see 7.3.5).

The Accept header field shall specify media types identifying CIM-RS payload representations (including version) that are supported by the client.

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

NOTE RFC2616 permits the use of media ranges for the Accept header field. However, with the envisioned combinations of type and subtype values for CIM-RS, wildcarding based on type and subtype is not meaningful.

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

NOTE RFC2616 does not specify recommendations for implementing the "q" accept parameter.

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

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

A server shall use one of the payload representations and versions identified in the Accept header field for the response payload, considering the "q" accept parameter if implemented.

The payload representation version specified in the media type (see 9.1) shall be interpreted by the server as follows:

- The update version is optional to be included. If an update version is included, it specifies the lowest acceptable update version (within the specified major version and acceptable minor versions); higher update versions shall be acceptable in addition. If no update version is included, the server shall assume a default of 0; that is, any update version is acceptable (within the specified major version and acceptable minor versions).
- The minor version is required to be included and specifies the only acceptable minor version.
- The major version is required to be included and specifies the only acceptable major version.

NOTE These rules follow the usual DMTF convention for referencing versions: Update versions newer than the one specified are selected automatically if available, but newer minor (and of course, major) versions are not selected automatically.

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

NOTE RFC2616 only recommends returning HTTP status code 406 "not acceptable" in this case, but it does not require it.

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

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

The following example assumes a JSON-based payload representation identified by `"application/json"` and an XML-based payload representation identified by `"text/xml"`. Actual payload representations may define different media types.

Example:

```
Accept: application/json; version=2.0,
        application/json;version=2.0.1; q=0.5,
        text/xml; version=2.0;q=0.2
```

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

8.4.2 Content-Type

The rules for the Content-Type entity-header field defined in RFC2616 apply. This subclause defines additional constraints on its use.

As defined in RFC2616, the Content-Type entity-header field shall be provided on the request message of any operation that passes a request payload and on the response message of any operation that returns a response payload.

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

8.4.3 X-CIMRS-Version

The CIM-RS protocol version is the version of this document, without any draft level.

The X-CIMRS-Version extension-header field shall identify the CIM-RS protocol version to which the request or response conforms, using the following format for its field value (defined in ABNF):

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

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

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

Example:

```
X-CIMRS-Version: 2.0.0
```

9 Payload representation

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

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

9.1 Internet media types

The CIM-RS protocol uses Internet media types for identifying the payload representation of its abstract payload elements. This subclause defines requirements related to media types used for the CIM-RS protocol.

Each CIM-RS payload representation specification shall define a media type as defined in RFC6838 and RFC6839 that uniquely identifies its payload representation within the set of payload representations listed in Table 44, and that identifies the version of the payload representation (typically by using a media type parameter such as "version").

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

9.2 Payload element representations

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

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

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

For example, in a JSON representation of an Instance payload element (see 7.5.2), all of the following options would be valid for representing the "self" attribute for resource identifier "/machine/1234":

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

```
{
  "self": "/root%2Fcimv2/classes/ACME_ComputerSystem/instances/sys11",

  "self": {
    "href": "/root……",
    "classname": "..",
    "namespace": "..",
    "keys": { "key1": <like any property value>, …}
  }
  . . .
}
```

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

```
{
  "this": "/root%2Fcimv2/classes/ACME_ComputerSystem/instances/sys11",
  . . .
}
```

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

```
{
  "links": [
    { "rel": "self",
      "href": "/root%2Fcimv2/classes/ACME_ComputerSystem/instances/sys11" },
    . . .
  },
  . . .
}
```

9.3 Payload representations

Table 44 lists known payload representations for this major version of the CIM-RS protocol and requirements to implement them; payload representations not listed in Table 44 may be implemented in addition.

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

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

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

WBEM servers should implement the SLP discovery protocol, supporting the provisions set forth in DSP0205, and the SLP template defined in DSP0206.

The CIM-RS protocol has no requirements for supporting the discovery of listeners.

11 Version compatibility

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

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

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

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

- HTTP protocol (see 11.1)
- CIM-RS protocol (see 11.2)
- CIM-RS payload representation (see 11.3)

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

11.1 HTTP protocol version compatibility

As defined in RFC2616, every HTTP request and every HTTP response shall indicate the HTTP protocol version to which the message format conforms.

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

At this point, there is no HTTP version higher than 1.1 defined.

11.2 CIM-RS protocol version compatibility

As defined in 8.4.3, every HTTP request and every HTTP response in the CIM-RS protocol shall indicate the CIM-RS protocol version to which the request or response conforms, by including the X-CIMRS-Version extension-header field. As defined in 8.4.3, the X-CIMRS-Version extension-header field identifies major, minor and update version of the CIM-RS protocol.

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

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

The update version is not considered in this rule because new update versions (within the same major and minor version) are not supposed to introduce new functionality, so this rule allows clients and servers 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

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

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

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

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

12 Conformance

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

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

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

ANNEX A
(normative)

Common ABNF rules

This annex defines common ABNF rules used throughout this document.

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

decimalDigit = "0" / nonZeroDecimalDigit

leadingZeros = 1*"0"

positiveDecimalInteger = [leadingZeros] nonZeroDecimalDigit *decimalDigit

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

ANNEX B (normative)

Mapping CIM-RS to generic operations

This annex describes how CIM-RS operations shall be mapped to generic operations (see DSP0223). This mapping is useful when implementing the CIM-RS protocol in WBEM servers and listeners that internally support the semantics of generic operations.

B.1 Query parameters

Most of the CIM-RS query parameters (see 6.6) can be used with multiple CIM-RS operations. Likewise, many generic operations input parameters are common between multiple generic operations, and are used consistently across those operations. With minor exceptions, the usage of any particular CIM-RS query parameter can be mapped directly to specific generic operation parameters, regardless of the CIM-RS operation with which it is used.

Table 45 defines the mapping of CIM-RS query parameters to generic operations input parameters.

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

B.2 Server operations

This subclause describes a server's decision tree for how incoming CIM-RS operations shall be analyzed, identified, and mapped to generic operations. The server can determine the generic operation based on the HTTP method and the target resource type.

The target resource type can be identified from the format of the path component of the target resource identifier, as shown in Table 46.

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

The generic operation(s) that shall be invoked for each combination of HTTP method and resource type are shown in Table 47. The query parameters shall be mapped to generic operation parameters as described in Table 45; column "Generic Operation Parameters" in Table 47 lists additional constraints on generic operation parameters.

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

This subclause describes a listener's decision tree for how incoming CIM-RS listener operations shall be analyzed, identified, and mapped to generic listener operations.

The listener can determine the generic operation based on the HTTP method and the target resource type.

The target resource type can be identified from the format of the path component of the target resource identifier, as shown in Table 48.

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

The generic operation(s) that should be invoked for each combination of HTTP method and resource type are shown in Table 49. The query parameters are mapped to generic operation parameters as described in Table 45; column "Generic Operation Parameters" in Table 49 lists additional constraints on generic operation parameters.

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

ANNEX C
(normative)

Mapping generic operations to CIM-RS

This annex describes how generic operations (see DSP0223) are mapped to CIM-RS operations. This mapping is provided primarily to describe how the CIM-RS protocol conforms to generic operations. This mapping is also useful for translating operation requirements defined in management profiles that are stated in terms of generic operations, into CIM-RS operations.

C.1 Conformance

CIM-RS does not satisfy all conformance requirements defined in generic operations (DSP0223). As a result, CIM-RS is not a conforming WBEM protocol. The remaining subclauses in this annex provide details.

C.2 Support of optional generic operations features

This subclause describes how CIM-RS supports optional features defined in generic operations.

- CIM-RS does not support the exclusion of all inherited properties and methods with one parameter when retrieving classes (that is, the equivalent of generic operation parameter `IncludeInheritedElements=False`).
- CIM-RS supports the inclusion of specific properties when retrieving instances (that is, the equivalent of generic operation parameter `IncludedProperties`)
- CIM-RS supports the specification of initial property values when creating an instance (that is, the equivalent of generic operation parameter `NewInstance`)
- CIM-RS supports error handling by means of returning DMTF standard messages (also known as "extended error handling")
- CIM-RS supports filter queries in pulled instance operations (that is, the equivalent of generic operation parameter `FilterQueryString`). The DMTF *Filter Query Language* (see DSP0212) is required to be supported as a query language. Other query languages are not currently supported with CIM-RS.
- CIM-RS supports client side control of continuation on error for pulled instance enumeration operations (that is, the equivalent of generic operation parameter `ContinueOnError`)

C.3 Operations

This subclause describes how the generic operations are supported in CIM-RS.

C.3.1 Server operations

The generic server operations listed in Table 47 are supported as described there.

Table 50 lists generic server operations that are not supported in CIM-RS. These operations are the 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.

ANNEX D (informative)

Change log

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)

Bibliography

This annex contains a list of informative references for this document.

DMTF DSP0200, *CIM Operations over HTTP 1.3*, http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf

DMTF DSP1001, *Management Profile Specification Usage Guide 1.1*, http://www.dmtf.org/standards/published_documents/DSP1001_1.1.pdf

DMTF DSP1033, *Profile Registration Profile 1.1*, http://www.dmtf.org/standards/published_documents/DSP1033_1.1.pdf

DMTF DSP1054, *Indications Profile 1.2*, http://www.dmtf.org/standards/published_documents/DSP1054_1.2.pdf

DMTF DSP1092, *WBEM Server Profile 1.0*, http://www.dmtf.org/standards/published_documents/DSP1092_1.0.pdf

DMTF DSP2032, *CIM-RS White Paper 1.0*, http://www.dmtf.org/standards/published_documents/DSP2032_1.0.pdf

IETF RFC2608, *Service Location Protocol, Version 2*, June 1999, http://tools.ietf.org/html/rfc2608

IETF RFC4648, *The Base16, Base32, and Base64 Data Encodings*, October 2006, http://tools.ietf.org/html/rfc4648

IETF RFC5005, *Feed Paging and Archiving*, September 2007, http://tools.ietf.org/html/rfc5005

IETF RFC7159, *The JavaScript Object Notation (JSON) Data Interchange Format*, March 2014, http://tools.ietf.org/html/rfc7159

IANA Permanent Message Header Field Names, http://www.iana.org/assignments/message-headers/perm-headers.html

IANA MIME Media Types, http://www.iana.org/assignments/media-types/

IANA Port Number Registry http://www.iana.org/assignments/port-numbers

ITU-T X.509, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*, http://www.itu.int/rec/T-REC-X.509/en

R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, PhD thesis, University of California, Irvine, 2000, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

R. Fielding, *REST APIs must be hypertext driven*, October 2008, http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

J. Holzer, *RESTful Web Services and JSON for WBEM Operations*, Master thesis, University of Applied Sciences, Konstanz, Germany, June 2009, http://mond.htwg-konstanz.de/Abschlussarbeiten/Details.aspx?id=1120

A. Manes, *Rest principle: Separation of representation and resource*, March 2009, http://apsblog.burtongroup.com/2009/03/rest-principle-separation-of-representation-and-resource.html

L. Richardson and S. Ruby, *RESTful Web Services*, May 2007, O'Reilly, ISBN 978-0-596-52926-0, http://www.oreilly.de/catalog/9780596529260/