Document Number: DSP0210

Date: 2013-01-24

Version: 1.0.0

CIM-RS Protocol

Document Type: Specification

Document Status: DMTF Standard

Document Language: en-US

CONTENTS

Figures

Tables

Foreword

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

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
- George Ericson, EMC
- Johannes Holzer, IBM
- Robert Kieninger, IBM
- Wojtek Kozaczynski, Microsoft
- Larry Lamers, VMware
- Andreas Maier, IBM (editor)
- 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 and additions:

- 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.
- The hash character "#" is used to denote a comma separated list of the rule following the hash character (similar to how "*" indicates a list of the rule following it, just without separator characters). The separator comma may be surrounded by linear whitespace, empty list items (that is, comma followed by comma) get eliminated, and multiplicity modifiers are supported, as described for "#rule" in section 2.1 of RFC2616.

The following general ABNF rules are defined:

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

Experimental material

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

The following typographical convention indicates experimental material:

EXPERIMENTAL

Experimental material appears here.

EXPERIMENTAL

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

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.

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 2.7*,
http://www.dmtf.org/standards/published_documents/DSP0004_2.7.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 DSP0212, *Filter Query Language 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0212_1.0.pdf

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

DMTF DSP0211, *CIM-RS Payload Representation in JSON 1.0*, http://www.dmtf.org/standards/published_documents/DSP0211_1.0.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

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 DSP0004 and DSP0223 apply to this document. Specifically, this document uses the terms "namespace", "qualifier", "qualifier type", "class", “creation class”, "ordinary class", "association", "indication", "instance", "property", "ordinary property", "reference", "method", "parameter", and "return value" defined in DSP0004.

The following additional terms are used in this document.

3.1

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.

3.2

CIM-RS payload element

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

3.3

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.

3.4

CIM-RS payload representation specification

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

3.5

CIM-RS protocol

the protocol defined in this document and related documents.

3.6

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.

3.7
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.

3.8
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.

3.9
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.

3.10
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.

3.11
HTTP entity body

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

3.12
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".

3.13
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".

3.14
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".

3.15
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.

3.16
HTTP message

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

3.17
HTTP method

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

3.18

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".

3.19

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".

3.20

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".

3.21

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".

3.22

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.

3.23

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.

3.24

method invocation link

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

3.25

model

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

3.26

navigation property

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

3.27

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.*

3.28

reference-typed parameter

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

3.29

reference-typed property

a CIM property declared with a CIM datatype that is a reference to a specific class. See 5.4.3 for details. DSP0004 defines the term "reference" for such properties; this document uses the more specific term "reference-typed property", instead.

3.30

reference-qualified property

a string-typed CIM property qualified with the *Reference* qualifier (see DSP0004 for a definition of the *Reference* qualifier, and 5.4.3 for details).

3.31

reference property

a general term for reference-typed properties and reference-qualified properties. See 5.4.3 for details.

3.32

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.

3.33

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*.

3.34

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".

3.35

WBEM client

the client role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see 5.1.

3.36

WBEM listener

the event listener role in the CIM-RS protocol and in other WBEM protocols.. For a full definition, see 5.1.

3.37

WBEM server

the server role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see 5.1.

4 Symbols and abbreviated terms

The abbreviations defined in DSP0004 and DSP0223 apply to this document. The following additional abbreviations are used in this document.

4.1
ABNF
Augmented Backus-Naur Form, as defined in RFC5234.

4.2
CIM
Common Information Model, as defined by DMTF.

4.3
CIM-RS
CIM RESTful Services
the name of the protocol defined in this document and related documents.

4.4
FQL
Filter Query Language, as defined by DMTF.

4.5
HTTP
Hyper Text Transfer Protocol. HTTP version 1.1 is defined in RFC2616. Unless otherwise noted, the term HTTP is used in this document to mean both HTTP and HTTPS.

4.6
HTTPS
Hyper Text Transfer Protocol Secure, as defined in RFC2818.

4.7
IANA
Internet Assigned Numbers Authority; see http://www.iana.org.

4.8
JSON
JavaScript Object Notation, as defined in ECMA-262.

4.9
REST
Representational State Transfer, as originally and informally described in *Architectural Styles and the Design of Network-based Software Architectures*.

4.10
SLP
Server Location Protocol, as defined in RFC2608.

4.11
UCS
Universal Character Set, as defined in ISO/IEC 10646:2003.

4.12
URI

Uniform Resource Identifier, as defined in RFC3986.

4.13
UTF-8

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

4.14
WBEM

Web Based Enterprise Management, as defined by DMTF.

4.15
XML

eXtensible Markup Language, as defined by W3C.

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): *operation*s 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.

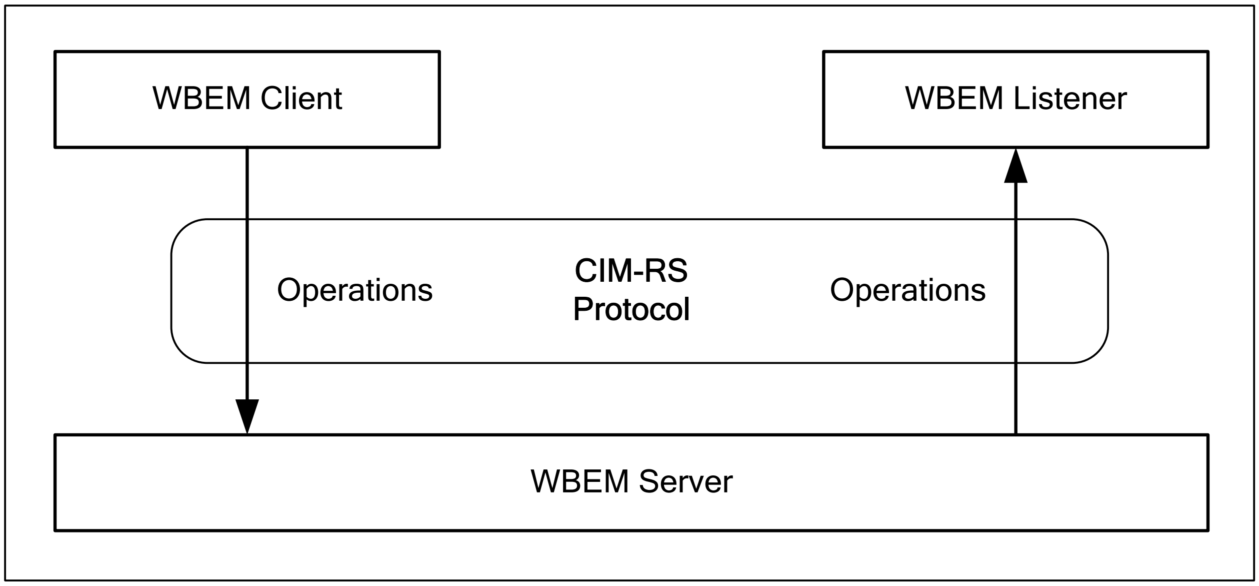

Figure 1 – Participants in the CIM-RS protocol

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.

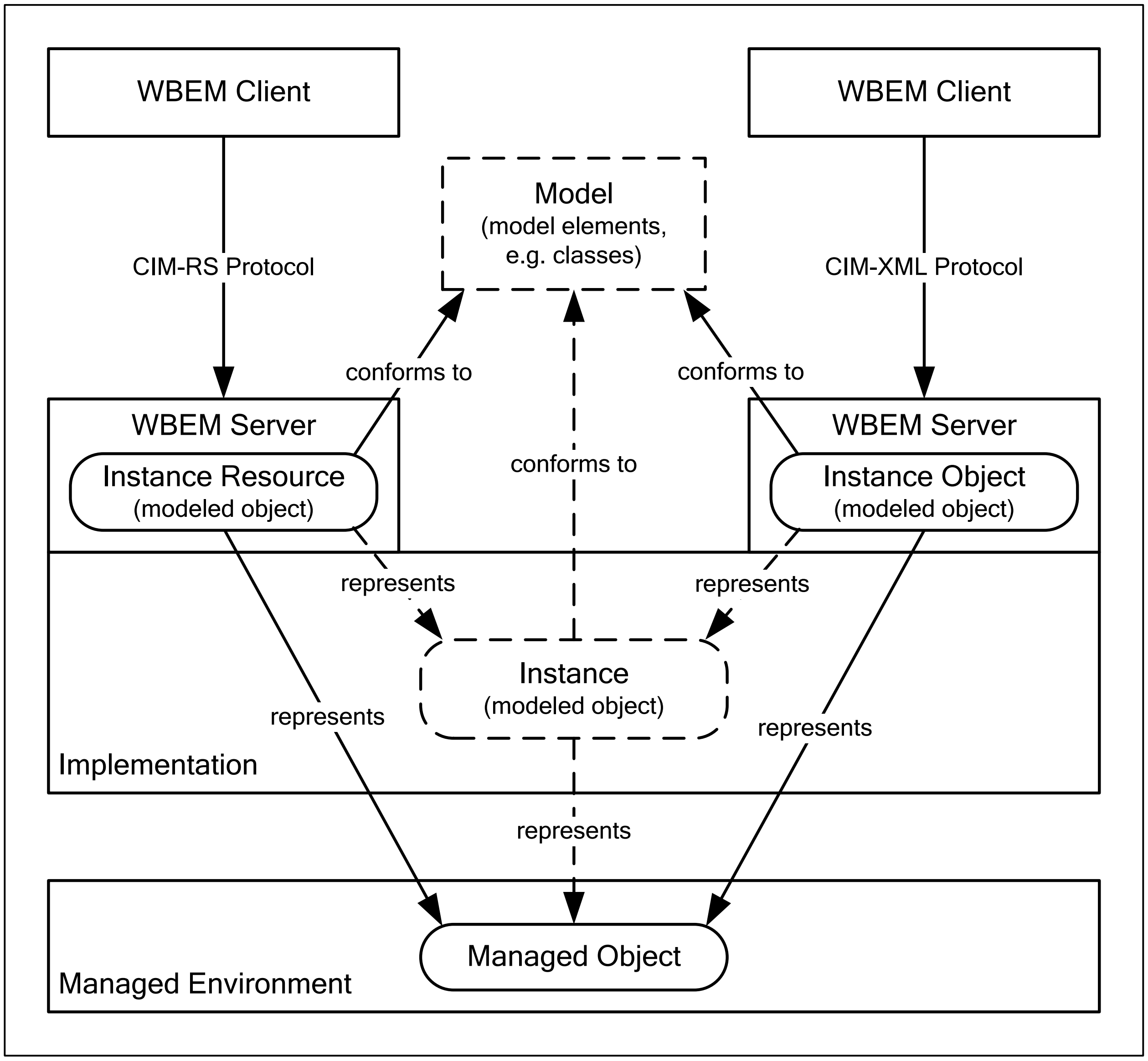

Figure 2 – Single model and multiple protocols

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 Basic kinds of resources

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

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

5.4 Mapping model elements to CIM-RS resources

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

5.4.1 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.

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

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

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

5.4.2 Instances

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

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 specific invocation resources related to the instance resource (see 5.4.4).

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.

5.4.3 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 datatype 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 datatype in Table 5.

5.4.4 Methods and operations

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

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

5.5 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:

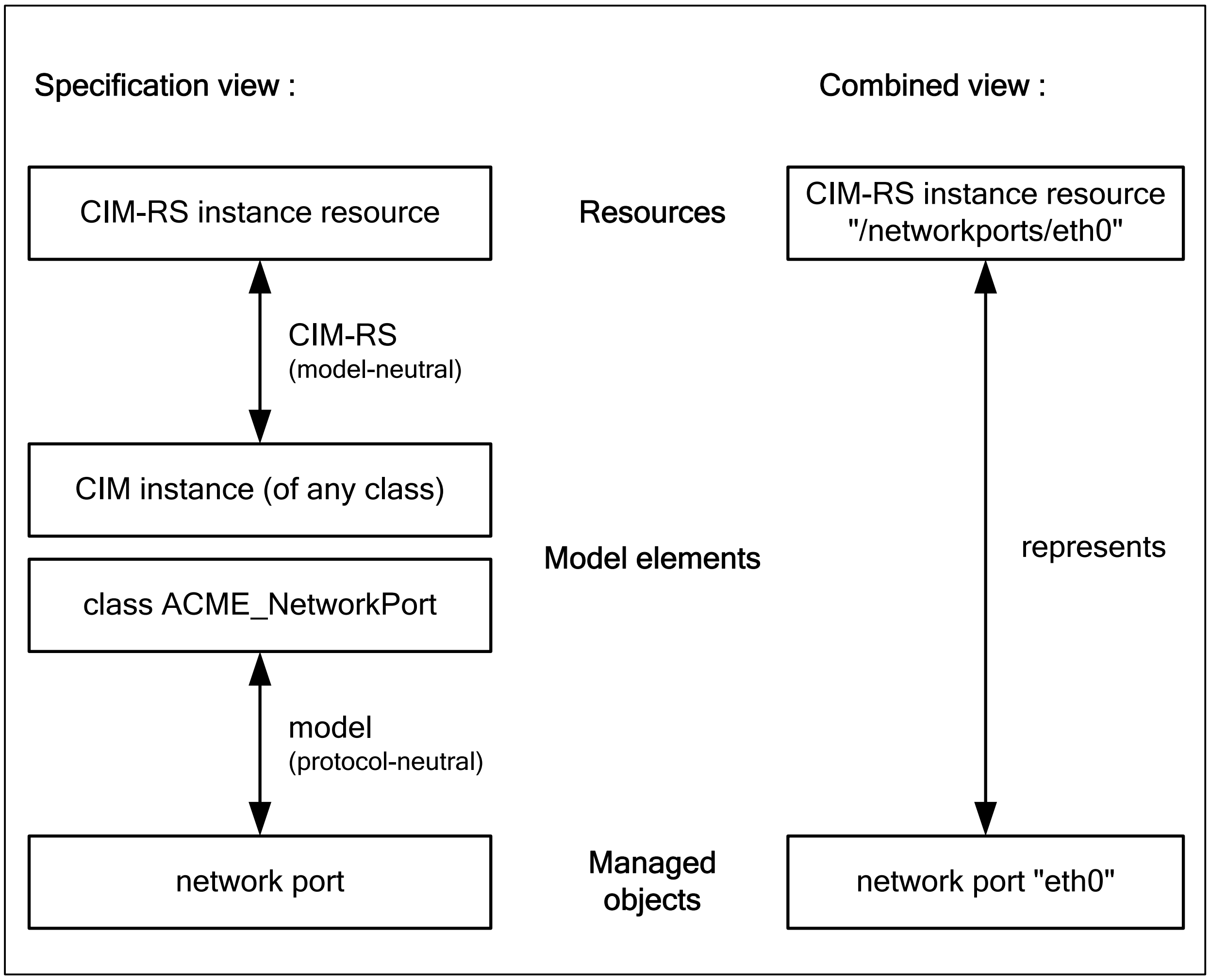

Figure 3 – Two-staged mapping approach in CIM-RS

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.

5.6 Navigation between resources

Clients can navigate between resources in any of these ways:

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

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

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

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

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

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

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

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

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

```
nav-path = nav-hop *( "." nav-hop )

nav-hop = nav-filter  ( embedded-path  ref-name / assoc-class-name )

embedded-path = *( prop-name "." )

nav-filter = ( "[" filter-class-name "]" )
```

Where:

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

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

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

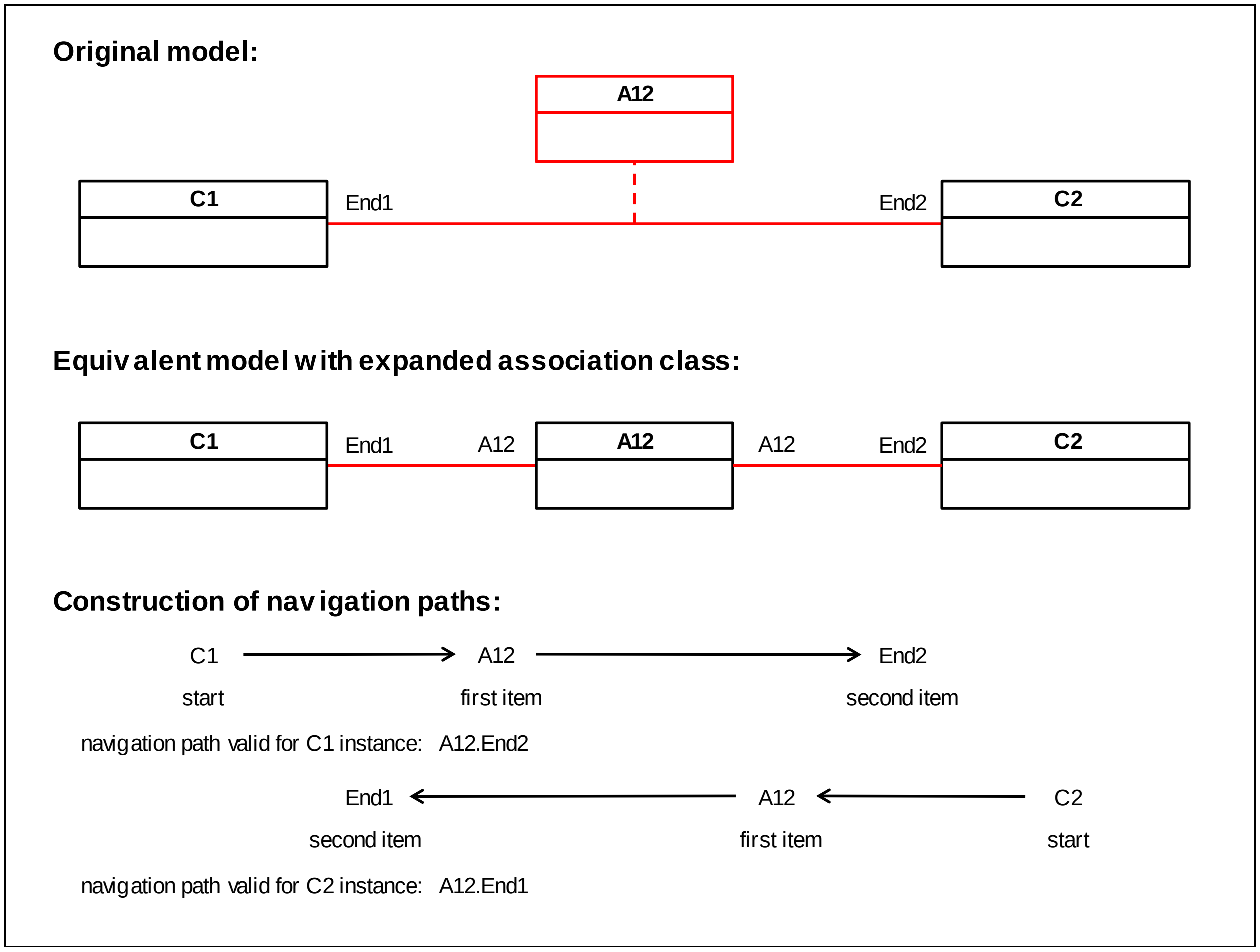
Original model:
A12
C1
End1
End2
C2
Equivalent model with expanded association class:
C1
End1
A12
A12
A12
End2
C2
Construction of navigation paths:
C1
A12
End2
start
first item
second item
navigation path valid for C1 instance: A12.End2
End1
A12
C2
second item
first item
start
navigation path valid for C2 instance: A12.End1

Figure 4 – Expanding association classes to construct navigation paths

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

Navigation paths identify their target instances as follows:

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

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

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

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

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

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

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

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

5.7 Discovering resources in a server

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

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

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

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

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

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

5.8 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.

In CIM-RS, resources are addressed through resource identifiers that are URIs. 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 the recommendations that all addressing information for a resource is in the resource identifier and on opaqueness and non-assembly of the resource identifier.

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.

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

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

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

Where:

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

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

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

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).

6.2 Opaqueness

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

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

- Construction of the resource identifier for the server entry point resource
- Parsing, adding, removing or modifying any query parameters in the resource identifier
- Normalizing the resource identifier, as described in RFC3986 (for example, removing ".." and "." segments)

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

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

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

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

considered delimiter or data in that query parameter, or sometimes even within portions of the query parameter.

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

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

The definitions of the query parameters in 6.5 defines 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 with CIM datatypes need to be represented in resource identifiers, the datatype-specific string representations defined in DSP0004 should 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 we assume that the usage of the namespace name has defined 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 allowed in resource identifiers 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.

6.4 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 HTTP and HTTPS:
  - port number 80 when using HTTP
  - port number 443 when using HTTPS

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).

6.5 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)

Example:

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

  This resource identifier specifies the query parameters $filter with a value of `Name='eth0'` and $properties with a value of `Name,Description`
- `/cimrs/networkports?$filter=Description='a%26b'`

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

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 shall be supported, subject to the usage constraints defined in this document:

Table 1 – Query parameters in CIM-RS

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

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

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.

6.5.1 $class (specify class name)

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

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)
- `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:

```
$class=ACME_ComputerSystem
```

specifies class name ACME_Computersystem

6.5.2 $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 7.3.8.

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)

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 without a value or 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

`$continueonerror`
`$continueonerror=true`

The server continues paged retrieval sequences in case of errors

6.5.3 $expand (include target instances)

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

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

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

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

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

```
query-parameter =/ expand-query-parm

expand-query-parm = "$expand=" [ expand-list ]

expand-list = nav-path *( "," nav-path )
```

Where:

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

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

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

Examples:

(not specified)
`$expand=`

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

`$expand=ACME_SystemDevice.PartComponent`

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

`$expand=Volumes`

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

For more examples, see D.1.

6.5.4 $filter (filter instances in result)

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

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

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

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

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

The $filter query parameter may be repeated in a resource identifier, see 6.5. 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.

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

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:

(not specified)

no additional restrictive instance filtering takes place

`$filter=`

includes no instances

`$filter=Type='LAN'%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 reserved characters "=" and single quote (') in the FQL query string are not percent-encoded because they do not occur in literals of the FQL query string and are therefore considered delimiters.

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

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

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

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

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

6.5.5 $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 7.3.8.

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)
- `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 for the number of members in the collection result.

`$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.

6.5.6 $methods (subset method links in result)

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

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

```
query-parameter =/ methods-query-parm

methods-query-parm = "$methods=" [ method-list ]

method-list = method-spec *( "," method-spec )

method-spec = [ nav-path "." ] method-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). Note that the character "." used in the in the literals of these ABNF rules is an unreserved character.
- `method-name` is the name of a method (without parenthesis or any method parameters)
- `nav-path` is the navigation path to the instances whose method invocation links are to be subsetted. `nav-path` and the concept of a navigation path is described in 5.6. Any reserved characters in the navigation path (that is, "[" and "]") shall be considered delimiters for purposes of percent-encoding (see 6.3). Note that the character "." in the navigation path is an unreserved character.

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

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

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

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 the instance collection itself and to all instances in the collection.

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

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

Examples:

(not specified)

no method invocation links are excluded

`$methods=`

no method invocation links are included

`$methods=Start,Stop`

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

6.5.7 $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 7.3.8.

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)
- `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 the default paging timeout of the server (see 7.12).

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

Examples:

(not specified)

default paging timeout of the server is used

`$pagingtimeout=0`

no paging timeout is used (infinite paging timeout)

`$pagingtimeout=30`

a paging timeout of 30 seconds is used

6.5.8 $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-spec *( "," property-spec )

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

The $properties query parameter may be repeated in a resource identifier, see 6.5. 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.

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

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

6.5.9 $refer (include references to target instances)

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

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

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

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

```
query-parameter =/ refer-query-parm

refer-query-parm = "$refer=" [ refer-list ]

refer-list = nav-path *( "," nav-path )
```

Where:

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

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

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

Examples:

(not specified)

`$refer=`

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

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

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

For more examples, see D.1.

6.6 Resource identifiers of entry point resources

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

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

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

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

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

Examples:

`/cimrs`

`http://acme.com/cimrs/`

7 Resources, operations and payload elements

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

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.

Table 2 – Resource types in CIM-RS

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

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

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

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

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 7.6.1
ReferenceCollection	representation of a reference collection resource containing an order-preserving list of references to instance resources	See 7.7.1

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

7.2 Description conventions

7.2.1 Datatypes used in payload element definitions

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

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

Table 5 – Datatypes used in payload elements

Payload datatype	Description
String	a string of UCS characters, or Null
Integer	an integer value, or Null
MethodLink	a complex type for method invocation links, containing the following child attributes:

Attribute	Payload datatype	Description
name	String	name of the method (without any parenthesis or method parameters)
class	String	name of the implemented class exposing the method
uri	URI	resource identifier of the method invocation resource (see 7.10)

Payload datatype	Description
ElementValue	a complex type for representing the value of a typed CIM element (such as properties, method parameters or method return values), and optionally its CIM datatype, containing the following child attributes:

Attribute	Payload datatype	Description
name	String	name of the element
value	multiple	value of the element, represented as defined by the payload representation specification. Reference properties and reference parameters need to be represented as defined for the URI payload datatype.
type	String	identification of the CIM datatype of the element, using the type strings defined by the payload representation specification

Payload datatype	Description
URI	a CIM-RS resource identifier, in the format defined in 6.1
Instance	an Instance payload element, as defined in 7.6.1

The CIM datatype 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 datatypes. This is expected to be used for WBEM client implementations as model-neutral client libraries.

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

7.2.2 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.

7.2.3 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. It is not expected that the implementation of the operation is specific to a class or model.

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

7.2.4 CIM-RS operation description format

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

Name:	The name of the operation.
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 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 case Other response headers shall not be specified by the responder. If other query request headers are specified by the responder, they shall be ignored by the requester, in order to provide for future extensibility.
Response payload:	The name of the payload element that shall be used in the entity body of the response message, separately for the success and failure case. "None" means the entity body shall be empty.
Requirement:	The requirement level to implement the operation, as defined in 7.2.3.
Description:	A normative definition of the behavior of the operation, in addition to the normative definitions stated in the previous description fields.
Example HTTP conversation:	An example HTTP request and HTTP response.

7.3 Common behaviors for all operations

7.3.1 Content negotiation

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

7.3.2 Verifying the basis of resource modifications (EXPERIMENTAL)

EXPERIMENTAL

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

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

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

EXPERIMENTAL

7.3.3 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.

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

7.3.4 Success and failure

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

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 7.3.6) and no 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.

7.3.5 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.

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

7.3.6 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 6 – Attributes of an ErrorResponse payload element

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

7.3.7 Consistency model

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

7.3.8 Paging of collections

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

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

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

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

If the server uses paging for a collection, its "next" attribute shall reference a collection resource that contains the next subset of collection members. 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.

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

As a result, any InstanceCollection payload element (see 7.8.1) or ReferenceCollection payload element (see 7.7.1) 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 instances into subset collections always happens at a granularity of complete instances (that is, instances are never broken apart to be returned in separate subset collections).

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

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

Table 7 – Operations supporting paging of collections

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

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

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

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

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

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

The first subset collection of a sequence shall not cease existence as a result of being successfully retrieved, when the server implements ceasing of subset collection resources (however, it may cease existence for other reasons, such as ceasing of the represented managed object). Separate retrieval requests for the entire and first subset collection 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 first subset collection opens a new sequence of paged retrievals for the remaining subset collections.

Clients and servers may support the continue on error feature (see 7.4.2). Clients that support the continue on error feature may request continuation on error for paged retrievals by specifying the $continueonerror query parameter (see 6.5.2). If a retrieval request results in an error, the client has request 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 6.5.7) 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 default shall be used, which is indicated in the "defaultPagingTimeout" attribute of the server entry point resource (see 7.12). For details on this behavior, see the description of "operation timeout" of pulled enumerations in DSP0223.

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. paging 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 subsequent (second to last) subset collections. 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 Entity tagging feature

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

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

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

7.4.2 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 7.3.8.

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

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

7.5 Instance creation resource

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

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

7.5.1 POST

Purpose:	Creates an instance resource
HTTP method:	POST
Target resource:	Instance creation resource (see 7.5)
Query parameters:	$class
Request headers:	Host, Content-Length, Content-Type, X-CIMRS-Version
Request payload:	Instance (see 7.6.1), without the "self" and "methods" attributes
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 7.3.6)
Requirement:	Mandatory (class specific)

Description:

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

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

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

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.

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

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

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

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

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

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

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

Example HTTP conversation (using JSON):

Request:

```
POST /cimrs/root%2Fcimv2/create?class=ACME_RegisteredProfile HTTP/1.1
Host: server.acme.com:5988
Content-Length: XXX
Content-Type: application/json;version=1.0.0
X-CIMRS-Version: 1.0.0

{
  "kind": "instance",
  "class": "ACME_RegisteredProfile",
  "properties": {
    "RegisteredName": "Fan",
    "RegisteredOrganization": 2,
    "RegisteredVersion": "1.1.0"
  }
}
```

Response:

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

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

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

7.6 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, this defines how managed objects are represented as REST instance resources. For details, see 5.5.

7.6.1 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 8.

Table 8 – Attributes of an Instance payload element

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

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

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

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

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

7.6.2 DELETE

Purpose:	Deletes an instance resource
HTTP method:	DELETE
Target resource:	Instance resource (see 7.6)
Query parameters:	None
Request headers:	Host, 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 7.3.6)
Requirement:	Mandatory (class specific)

Description:

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

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

- 204 "No Content"

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

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

Example HTTP conversation (using JSON):

Request:

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

Response:

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

7.6.3 GET

Purpose:	Retrieves an instance resource
HTTP method:	GET
Target resource:	Instance resource (see 7.6)

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

Description:

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

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

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

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

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

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

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

Example HTTP conversation (using JSON):

Request:

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

Response:

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

```
Content-Length: XXX
Content-Type: application/json;version=1.0.1
X-CIMRS-Version: 1.0.1

{
  "kind": "instance",
  "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0",
  "class": "ACME_RegisteredProfile",
  "properties": {
    "InstanceID": "DMTF:Fan:1.1.0",
    "RegisteredName": "Fan",
    "RegisteredOrganization": 2,
    "RegisteredVersion": "1.1.0",
    . . .
  },
  "methods": {
    "GetCentralInstances": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3
A1.1.0/GetCentralInstances"
  }
}
```

7.6.4 PUT

Purpose:	Modifies an instance resource (partially or fully)
HTTP method:	PUT
Target resource:	Instance resource (see 7.6)
Query parameters:	$properties
Request headers:	Host, Content-Length, Content-Type, If-Match (EXPERIMENTAL), X-CIMRS-Version
Request payload:	Instance (see 7.6.1)
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 7.3.6)
Requirement:	Mandatory (class specific)

Description:

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

Partial modification of an instance is achieved by specifying the desired subset of properties in the resource identifier using the $properties query parameter (see 6.5.8). Since 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 $properties query parameter that are not properties of the target instance shall cause the server to fail the operation with HTTP status code 404 "Not Found". Properties specified in the $properties query parameter that are not mutable shall cause the server to fail the operation with HTTP status code 403 "Forbidden".

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

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

EXPERIMENTAL

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

EXPERIMENTAL

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

- 204 "No Content"

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

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

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

Request:

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

{
  "kind": "instance",
  "class": "ACME_RegisteredProfile",
  "properties": {
    "RegisteredName": "Fan",
    "RegisteredOrganization": 2,
    "RegisteredVersion": "1.1.1",
    "Caption": "A changed caption"
  }
}
```

Response:

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

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

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

Request:

```
PUT /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0?properties=Regist
eredVersion,Caption HTTP/1.1
Host: server.acme.com:5988
Content-Length: XXX
Content-Type: application/json;version=1.0.0
X-CIMRS-Version: 1.0.0

{
  "kind": "instance",
  "class": "ACME_RegisteredProfile",
  "properties": {
    "RegisteredVersion": "1.1.1",
    "Caption": "A changed caption"
  }
}
```

Response:

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

```
X-CIMRS-Version: 1.0.1
```

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

7.7 Reference collection resource

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

7.7.1 ReferenceCollection payload element

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

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

Table 9 – Attributes of an ReferenceCollection payload element

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

7.7.2 GET

Purpose:	Retrieves a reference collection resource
HTTP method:	GET
Target resource:	Reference collection resource (see 7.7)
Query parameters:	$max, $continueonerror, $pagingtimeout
Request headers:	Host, Accept, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (success):	ReferenceCollection (see 7.7.1)
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.6)

Requirement: Mandatory (class specific)

Description:

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

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

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

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

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

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

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

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

Example HTTP conversation (using JSON):

Request:

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

Response:

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

{
  "kind": "referencecollection",
  "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/refer/ACME
_ElementConformsToProfile.ManagedElement/part/2",
  "class": "ACME_Fan",
  "references": [
    "/cimrs/root%2Fcimv2/ACME_Fan/fan11",
    "/cimrs/root%2Fcimv2/ACME_Fan/fan12"
  ]
}
```

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

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

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

7.8 Instance collection resource

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

7.8.1 InstanceCollection payload element

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

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

Table 10 – Attributes of an InstanceCollection payload element

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

7.8.2 GET

Purpose:	Retrieves the next page of a paged instance collection resource
HTTP method:	GET
Target resource:	Page of an instance collection resource (see 7.8)
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 7.8.1)
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.6)
Requirement:	Mandatory (class specific)

Description:

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

The target resource identifier for this operation is discovered from the "next" attribute of the previous page of the instance collection (see 7.3.8).

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

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

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

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

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

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

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

Example HTTP conversation (using JSON):

Request:

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

Response:

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

{
  "kind": "instancecollection",
  "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/ACME_Refer
encedProfile/Antecedent",
  "class": "ACME_RegisteredProfile",
  "instances": [
    {
      "kind": "instance",
      "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0",
      "class": "ACME_RegisteredProfile",
      "properties": {
        "InstanceID": "DMTF:Fan:1.1.0",
        "RegisteredName": "Fan",
        "RegisteredOrganization": 2,
        "RegisteredVersion": "1.1.0",
        . . .,
        "ACME_ReferencedProfile": {
          "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/AC
ME_ReferencedProfile",
          "Dependent": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0
.0/ACME_ReferencedProfile/Dependent"
        }
      },
      "methods": {
        "GetCentralInstances": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AF
an%3A1.1.0/GetCentralInstances"
      }
    },
    . . .
  ]
}
```

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

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

7.9 Instance enumeration resource

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

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

7.9.1 GET

Purpose:	Enumerates instance resources by class
HTTP method:	GET
Target resource:	Instance enumeration resource (see 7.9)
Query parameters:	$class, $filter, $expand, $refer, $properties, $methods, $max, $continueonerror, $pagingtimeout
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 7.8.1)
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.6)
Requirement:	Mandatory (class specific)

Description:

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

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

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

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

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

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

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

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

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

Example HTTP conversation:

Request:

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

Response:

```
HTTP/1.1 200 OK
Date: Fri, 11 Nov 2011 10:11:00 GMT
Content-Length: XXX
Content-Type: application/json;version=1.0.0
X-CIMRS-Version: 1.0.0

{
  "kind": "instancecollection",
  "self": "/cimrs/root%2Fcimv2/enum?class=ACME_System",
  "class": "ACME_System",
  "instances": [
    {
      "kind": "instance",
      "self": "/cimrs/root%2Fcimv2/ACME_ComputerSystem/sys1",
      "class": "ACME_ComputerSystem",
      "properties": {
        "InstanceID": "sys1",
        "Name": "sys1",
        . . .
      },
      "methods": {
        "RequestStateChange": "/cimrs/root%2Fcimv2/ACME_ComputerSystem/sys1/Request
StateChange"
      }
    },
    . . .
  ]
}
```

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

7.10 Method invocation resource

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

7.10.1 MethodRequest payload element

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

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

Table 11 – Attributes of a MethodRequest payload element

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

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

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

7.10.2 MethodResponse payload element

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

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

Table 12 – Attributes of a MethodResponse payload element

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

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

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

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

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

7.10.3 POST

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

Description:

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

The method can be static or non-static:

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

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

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

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

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

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

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

Request:

```
POST /cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentralInstan
ces HTTP/1.1
Host: server.acme.com:5988
Accept: application/json;version=1.0
Content-Length: XXX
Content-Type: application/json;version=1.0.0
X-CIMRS-Version: 1.0.0

{
  "kind": " methodrequest",
  "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentral
Instances",
  "method": "GetCentralInstances",
  "parameters": {
    "MaxNumber": 1000
  }
}
```

Response:

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

```
X-CIMRS-Version: 1.0.1

{
  "kind": " methodresponse",
  "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentral
Instances",
  "method": "GetCentralInstances",
  "returnvalue": 0,
  "parameters": {
    "ActualNumber": 25
  }
}
```

7.11 Listener destination resource

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

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

7.11.1 IndicationDeliveryRequest payload element

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

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

Table 13 – Attributes of an IndicationDeliveryRequest payload element

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

7.11.2 POST

Purpose:	Delivers an indication to a listener
HTTP method:	POST
Target resource:	Listener destination resource (see 7.11)
Query parameters:	None
Request headers:	Host, Accept, Content-Length, Content-Type, X-CIMRS-Version
Request payload:	IndicationDeliveryRequest (see 7.11.1)
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 7.3.6)

Requirement: Mandatory

Description:

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

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

- 200 "OK"

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

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

Example HTTP conversation (using JSON):

Request:

```
POST /cimrs/dest1 HTTP/1.1
Host: listener.acme.com:5988
Accept: application/json;version=1.0
Content-Length: XXX
Content-Type: application/json;version=1.0.0
X-CIMRS-Version: 1.0.1

{
  "kind": "indicationdeliveryrequest",
  "self": "/cimrs/dest1",
  "indication": {
    "kind": "instance",
    "class": "ACME_AlertIndication",
    "properties": {
      "AlertType": 4,
      "PerceivedSeverity": 5,
      "ProbableCause": 42,
      "Message": "BOND0007: Some error happened, rc=23.",
      "MessageArguments": [ "23" ],
      "MessageID": "BOND0007",
      "OwningEntity": "ACME"
    }
  }
}
```

Response:

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

7.12 Server entry point resource

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

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

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

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

7.12.1 ServerEntryPoint payload element

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

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

Table 14 – Attributes of a ServerEntryPoint payload element

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

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

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

Table 15 – Attributes of SEPNamespace payload datatype

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

7.12.2 GET

Purpose:	Retrieves the entry point resource of a server
HTTP method:	GET
Target resource:	Server entry point resource (see 7.12)

Query parameters:	None
Request headers:	Host, X-CIMRS-Version
Request payload:	None
Response headers (success):	Date, X-CIMRS-Version
Response payload (success):	ServerEntryPoint (see 7.12.1)
Response headers (failure):	Date, Content-Length, Content-Type, X-CIMRS-Version
Response payload (failure):	ErrorResponse (see 7.3.6)
Requirement:	Mandatory

Description:

The HTTP GET method on a server entry point resource retrieves a representation of the specified server entry point resource. The returned ServerEntryPoint payload element describes protocol-level capabilities of the server and its namespaces, such as supported protocol versions and supported payload representations, as well as resource identifiers for discovering further resources in the server and its namespaces.

On success, the entity body shall contain a ServerEntryPoint payload element (see 7.12.1) and one of the following HTTP status codes shall be returned:

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

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

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

Example HTTP conversation:

Request:

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

Response:

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

{
  "kind": "serverentrypoint",
  "self": "/cimrs",
  "namespaces": [
    { "name": "interop",
      "enumeration": "/cimrs/interop/enum",
      "creation": "/cimrs/interop/create",
      "staticmethod": "/cimrs/interop/static",
      "protocolversions": [ "1.0.0", "1.0.1" ],
      "contenttypes": [
        "application/json;version=1.0.0",
        "application/json;version=1.0.1",
        "text/xml;version=1.0.0" ]
    },
    { "name": "root/cimv2",
      "enumeration": "/cimrs/root%2Fcimv2/enum",
      "creation": "/cimrs/root%2Fcimv2/create",
      "staticmethod": "/cimrs/root%2Fcimv2/static",
      "protocolversions": [ "1.0.0", "1.0.1" ],
      "contenttypes": [
        "application/json;version=1.0.0",
        "application/json;version=1.0.1",
        "text/xml;version=1.0.0" ]
    }
  ],
  "entitytagging": true,
  "pagedretrieval": true,
  "defaultpagingtimeout": 300,
  "minimumpagingtimeout": 1,
  "maximumpagingtimeout": 600,
  "continueonerror": true
}
```

7.13 Listener entry point resource

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

7.13.1 ListenerEntryPoint payload element

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

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

Table 16 – Attributes of a ListenerEntryPoint payload element

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

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

7.13.2 GET

Purpose: Retrieves the entry point resource of a listener

HTTP method: GET

Target resource: Listener entry point resource (see 7.13)

Query parameters: None

Request headers: Host, X-CIMRS-Version

Request payload: None

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

Response payload (success): ListenerEntryPoint (see 7.13.1)

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

Response payload (failure): ErrorResponse (see 7.3.6)

Requirement: Mandatory

Description:

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

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

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

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

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

Example HTTP conversation (server to listener):

Request:

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

Response:

```
HTTP/1.1 200 OK
Date: Fri, 11 Nov 2011 10:11:00 GMT
Content-Length: XXX
Content-Type: application/json;version=1.0.0
X-CIMRS-Version: 1.0.0

{
  "kind": "listenerentrypoint",
  "self": "/cimrs",
  "destinations": [ "/cimrs/dest1", "/cimrs/dest2" ],
  "protocolversions": [ "1.0.0" ],
  "contenttypes": [
    "application/json;version=1.0.0" ]
}
```

7.14 CIM-RS resources to be exposed

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

7.14.1 Resources exposed by a server

The following resources shall be exposed once by a server:

- Server entry point resource (see 7.12)

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

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

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

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

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

7.14.2 Resources exposed by a listener

The following resources shall be exposed once by a listener:

- Listener entry point resource (see 7.13)

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

- Listener destination resource (see 7.11)

7.15 Other typical WBEM protocol functionality

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 describes how a number of such functionalities can be performed.

7.15.1 Server discovery

WBEM servers can be discovered as described in clause 10.

7.15.2 Discovery of server and listener entry point resources

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

7.15.3 Namespace discovery

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

7.15.4 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. The management profiles to which a server advertises conformance can be discovered by enumerating instances of the CIM_RegisteredProfile class in the Interop namespace using the HTTP GET method on the instance enumeration resource for the Interop namespace (see 7.9.1).

7.15.5 Schema inspection

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

7.15.6 Association traversal

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

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

7.15.7 Indication subscription

The CIM-RS protocol defines the HTTP POST method on listener destination resources (see 7.11.2) 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 1 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 may be provided on the request message of any operation that may return a response payload.

If provided by a client, 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.

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

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

- 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 specifies the only acceptable minor version.
- The major version specifies the only acceptable minor 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 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.

Example:

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

In this example, 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 CIM-RS JSON payload representation (by means of the default value of 1 for the "q" parameter), if that representation version is not available, then for version 1.0.x (x>=1) of the CIM-RS JSON representation, if that is not available then for version 1.0.x (x>=0) of the CIM-RS XML 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 "version" parameter of the media type shall include the major, minor and update version indicators.

8.4.3 ETag (EXPERIMENTAL)

EXPERIMENTAL

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

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

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

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

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

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

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

EXPERIMENTAL

8.4.4 If-Match (EXPERIMENTAL)

EXPERIMENTAL

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

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

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

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

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

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

EXPERIMENTAL

8.4.5 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: 1.0.0
```

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, as defined in section 3.7 of RFC2616, for identifying the payload representation of its abstract payload elements. This subclause defines requirements related to media types used for the CIM-RS protocol.

9.1.1 General

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

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

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

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

```
application/json; version=1.0
```

9.1.2 Media type parameters

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

Table 17 – Media type parameters

Parameter	Presence Requirement	Description
version	Mandatory	See 9.1.2.1.

9.1.2.1 Parameter "version"

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

```
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.

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

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

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 datatypes stated in Table 5 do not need to correspond 1:1 to datatypes 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 7.6.1), all of the following options would be valid for representing the "self" attribute for resource identifier "/cimrs/machine/1234":

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

```
{
  "self": "/cimrs/machine/1234",
  . . .
}
```

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

```
{
  "this": "/cimrs/machine/1234",
  . . .
}
```

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

```
{
  "links": [
    { "rel": "self",
      "href": "/cimrs/machine/1234" },
    . . .
  ],
  . . .
}
```

9.3 Payload representations

Table 18 lists known payload representations and requirements to implement them; payload representations not listed in Table 18 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 18 – 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, supporting the SLP template defined in DSP0206.

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

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. Therefore, a client and a server are compatible w.r.t. the HTTP protocol version only if they both support HTTP 1.1.

11.2 CIM-RS protocol version compatibility

As defined in 8.4.5, 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 8.4.5, 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
(informative)

Mapping CIM-RS to generic operations

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

B.1 URI composition

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

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

B.1.1 Instance creation resource

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

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

B.1.2 Instance resource

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

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

B.1.3 Page of instance or reference collection resource from association traversal

An instance collection resource represents a collection of instance resources (see 7.8). A reference collection resource represents a collection of references to instance resources (see 7.7). Instance or reference collection resources representing the result of an association traversal from a source instance do not have URIs; their representation is always embedded as the value of a navigation property (see 5.6) in the source instance. If such an instance or reference collection is returned using paging (see

7.3.8), the pages following the initial (embedded) part of the collection have URIs. The URI of such a page enables the server to identify:

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

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

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

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

B.1.5 Instance enumeration resource

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

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

B.1.6 Static method invocation resource

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

- CIM namespace of the class upon which the method is to be invoked;
- Name of the class upon which the method is to be invoked;
- Name of the method;
- The type of the resource (in this case, a static method invocation resource).

B.1.7 Non-static method invocation resource

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

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

B.1.8 Listener destination resource

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

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

B.1.9 Server and listener entry point resources

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

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

B.2 Query parameters

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

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

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

CIM-RS Query Parameter	Generic Operations Input Parameter	Mapping
`$class`		See individual operation/resource mappings in this annex
`$continueonerror`	`ContinueOnError`	Directly equivalent
`$expand`		See B.2.1
`$max`	`MaxObjectCount`	Directly equivalent
`$methods`	no equivalent	The `$methods` query parameter has no analog in generic operations because it only dictates what links will be included in the returned payload. Logic to implement the $methods query parameter will be confined to the server implementation's protocol handler and will not need to be passed on to providers or other server components.
`$pagingtimeout`	`OperationTimeout`	Directly equivalent
`$properties`	`IncludedProperties` and `ExcludeSubclassProperties`	`$properties` is set to contents of `IncludedProperties`; if `ExcludeSubclassProperties` is TRUE, list of properties is reduced by those defined in subclasses.
`$refer`		See B.2.1
`$filter`	`FilterQueryString` and `FilterQueryLanguage`	Directly equivalent. If `$filter` is specified, `FilterQueryString` is set to the `$filter` query parameter value; `FilterQueryLanguage` is set to `"DMTF:FQL"` (see C.2)

B.2.1 Special handling for $expand and $refer query parameters

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

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

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

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

An implementation may do the following.

1) Identify all association traversal paths identified in all `$expand` and `$refer` query parameters supplied to the current operation. Merge the paths into a tree representation, so that common early portions of the different traversal paths need not be redundantly traversed. In this way the instance targeted by the CIM-RS operation is applied to the root of the traversal tree, and the leaves of the traversal tree represent the results of the individual association traversal paths. Note that if some traversal paths are strict supersets of others, this will result in a situation where not all traversal paths end in leaf nodes of the traversal tree. For each instance targeted by the CIM-RS operation, the tree is traversed to identify and supply the additional information requested in the query parameters, as described in subsequent steps.

2) When `$expand` or `$refer` is supplied for any CIM-RS operation, it will map to generic operations in a common fashion regardless of which CIM-RS operation was invoked. In any case, it is assumed that the CIM-RS operation being invoked will begin by obtaining an initial instance or instance collection. Once that instance or collection is obtained, the following generic operations mapping will be performed, using the initial instance or instance collection as the “working instance collection”.

3) Obtain the initial association traversal element from the root of the traversal tree identified in step 1) above.

4) For each Working Instance in the working instance collection, perform the following. If the current traversal tree node specifies both association class and reference, then perform a generic operations `OpenAssociatedInstancePaths` operation; if only association class is given, perform a generic operations `OpenReferencingInstancePaths` operation. (See step 6) below for possible modifications to generic operations method being called.) In either case, the call is made with the following parameters:

   - `SourceInstancePath` is formed from:
     - The CIM namespace (extracted from the Working Instance);
     - The class name (extracted from the Working Instance);
     - Key property name/value pairs (extracted from the Working Instance).
   - `AssociationClassName` is extracted from the class name specified in the current traversal tree node.
   - `AssociatedClassName` is set to NULL.
   - `SourceRoleName` is set to NULL.
   - `AssociatedRoleName` is set to the reference name obtained from the current traversal tree node, if reference name is present; if not present, `AssociatedRoleName` is set to NULL.
   - `FilterQueryString` is set from the $filter query parameter as described in B.2.1.
   - `FilterQueryLanguage` is set to `"DMTF:FQL"` (see C.2).
   - `OperationTimeout` is set from the `$pagingtimeout` query parameter as described in Table B-1.
   - `ContinueOnError` is set from the `$continueonerror` query parameter as described in Table B-1.
   - `MaxObjectCount` is set from the `$max` query parameter as described in Table B-1.

5) If the current traversal tree node contains sub-nodes, then perform N recursions into step 4) above, setting the “current traversal tree node” to each of the N traversal tree sub-nodes.

6) Special case: if the current traversal tree node corresponds to a terminal node in a `$expand` query parameter, then entire instances must be obtained instead of only instance paths. Therefore:

   a) Call `OpenAssociatedInstacesWithPath` instead of `OpenAssociatedInstancePaths`, or

   b) Call `OpenReferencingInstancesWithPath` operation instead of `OpenReferencingInstancePaths`.

   c) In either case, the following parameters will be supplied to the generic operations method:

      - `IncludeClassOrigin` is set to FALSE.
      - `IncludedProperties` is set from the `$properties` query parameter as described in Table B-1.
      - `ExcludeSubclassProperties` is set to FALSE.

B.3 Server operations

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

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

B.3.1 POST instance creation resource

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

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

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

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

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

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

Restrictions: None.

B.3.2 POST static method invocation resource

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

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

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

- the `ClassPath` parameter is formed from CIM namespace and class name, which are formed from information units extracted from the target URI of the HTTP request (see B.1.6)
- the `MethodName` parameter is formed from information units extracted from the target URI of the HTTP request (see B.1.6)
- the `InParmValues` parameter is formed from the `parameters` attribute of the `MethodRequest` payload element in the HTTP request (see 7.10.1)

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

- the `OutParmValues` parameter is used to form the `parameters` attribute of the `MethodResponse` payload element in the HTTP response (see 7.10.2)
- the `ReturnValue` parameter is used to form the `returnvalue` attribute of the `MethodResponse` payload element in the HTTP response (see 7.10.2)

Restrictions: None.

B.3.3 POST non-static method invocation resource

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

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

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

- the `InstancePath` parameter is formed from CIM namespace, class name and key bindings, which are all formed from information units extracted from the target URI of the HTTP request (see B.1.7)
- the `MethodName` parameter is formed from information units extracted from the target URI of the HTTP request (see B.1.7)
- the `InParmValues` parameter is formed from the `parameters` attribute of the `MethodRequest` payload element in the HTTP request (see 7.10.1)

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

- the `OutParmValues` parameter is used to form the `parameters` attribute of the `MethodResponse` payload element in the HTTP response (see 7.10.2)
- the `ReturnValue` parameter is used to form the `returnvalue` attribute of the `MethodResponse` payload element in the HTTP response (see 7.10.2)

Restrictions: None.

B.3.4 DELETE instance resource

This CIM-RS operation deletes an instance resource (see 0).

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

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

- the `InstancePath` parameter is formed from CIM namespace, class name and key bindings, which are all formed from information units extracted from the target URI of the HTTP request (see B.1.7)

This generic operation has no output parameters.

Restrictions: None..

B.3.5 GET instance resource

This CIM-RS operation retrieves an instance resource (see 7.6.3), possibly including associated or referenced instance resources.

If neither the `$refer` nor the `$expand` query parameter is specified, this CIM-RS operation directly maps to the generic operation `GetInstance`.

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

- the `InstancePath` parameter is formed from CIM namespace, class name and key bindings, which are all formed from information units extracted from the target URI of the HTTP request (see B.1.2)
- the `IncludeClassOrigin` parameter is set to false
- the `IncludedProperties` parameter is obtained from the `$properties` query parameter as described in Table B-1

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

- the `Instance` parameter is used to form the `Instance` payload element in the HTTP response (see 7.6.1)

If the `$refer` or `$expand` query parameters are specified, this CIM-RS operation maps to the generic operation `GetInstance` as described above, and possibly additional association traversal operations, as described in B.2.1.

Restrictions:

- Including the class origin of properties in the returned instance representation is not supported in CIM-RS.

B.3.6 GET page of instance collection resource

This CIM-RS operation retrieves the next page of a paged instance collection resource (see 7.8.2), resulting from enumeration by class, or from association traversal.

This CIM-RS operation directly maps to the generic operation PullInstancesWithPath.

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

- the `NamespacePath` parameter is formed from the CIM namespace, which is formed from information units extracted from the target URI of the HTTP request (see B.1.3 and B.1.4)
- the `EnumerationContext` parameter is formed from the information about the next page to be retrieved within the overall collection, which is formed from information units extracted from the target URI of the HTTP request (see B.1.3 and B.1.4)

- the `MaxObjectCount` parameter is obtained from the `$max` query parameter as described in Table B-1

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

- the `InstanceList` parameter is used to form the `instances` attribute in the `InstanceCollection` payload element in the HTTP response (see 7.8.1)
- if the `EndOfSequence` parameter is FALSE, the `EnumerationContext` parameter is used to form the information about the next page to be retrieved within the overall collection, in the URI for the `next` attribute in the `InstanceCollection` payload element in the HTTP response (see 7.8.1)
- if the `EndOfSequence` parameter is TRUE, the `next` attribute is omitted from the `InstanceCollection` payload element in the HTTP response (see 7.8.1)

Restrictions: None.

B.3.7 GET page of reference collection resource

This CIM-RS operation retrieves the next page of a paged reference collection resource (see 7.7.2), resulting from enumeration by class, or from association traversal.

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

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

- the `NamespacePath` parameter is formed from the CIM namespace, which is formed from information units extracted from the target URI of the HTTP request (see B.1.3 and B.1.4)
- the `EnumerationContext` parameter is formed from the information about the next page to be retrieved within the overall collection, which is formed from information units extracted from the target URI of the HTTP request (see B.1.3 and B.1.4)
- the `MaxObjectCount` parameter is obtained from the `$max` query parameter as described in Table B-1

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

- the `InstancePathList` parameter is used to form the `references` attribute in the `ReferenceCollection` payload element in the HTTP response (see 7.7.1)
- if the `EndOfSequence` parameter is FALSE, the `EnumerationContext` parameter is used to form the information about the next page to be retrieved within the overall collection, in the URI for the `next` attribute in the `ReferenceCollection` payload element in the HTTP response (see 7.7.1)
- if the `EndOfSequence` parameter is TRUE, the `next` attribute is omitted from the `ReferenceCollection` payload element in the HTTP response (see 7.7.1)

Restrictions: None.

B.3.8 GET instance enumeration resource

This CIM-RS operation enumerates all instances of the specified class (including instances of subclasses) in the namespace of the targeted instance enumeration (see 7.9.1.

If neither the `$refer` nor the `$expand` query parameter is specified, this CIM-RS operation directly maps to the generic operation `OpenClassInstancesWithPath`.

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

- the `EnumClassPath` parameter is formed from:
  - the CIM namespace, formed from information units extracted from the target URI of the HTTP request (see B.1.5)
  - the class name, obtained from the $class query parameter in the target URI of the HTTP request (see B.1.5)
- the `FilterQueryString` parameter is set from the `$filter` query parameter as described in Table B-1
- the `FilterQueryLanguage` parameter is set to `"DMTF:FQL"` (see C.2)
- the `IncludeClassOrigin` parameter is set to false
- the `IncludedProperties` parameter is set from the `$properties` query parameter as described in Table B-1
- the `ExcludeSubclassProperties` parameter is set to false
- the `OperationTimeout` parameter is set from the `$pagingtimeout` query parameter as described in Table B-1
- the `ContinueOnError` parameter is set from the `$continueonerror` query parameter as described in Table B-1
- the `MaxObjectCount` parameter is set from the `$max` query parameter as described in Table B-1

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

- the `InstanceList` parameter is used to form the `instances` attribute in the `InstanceCollection` payload element in the HTTP response (see 7.8.1)
- if the `EndOfSequence` parameter is FALSE, the `EnumerationContext` parameter is used to form the information about the next page to be retrieved within the overall collection, in the URI for the `next` attribute in the `InstanceCollection` payload element in the HTTP response (see 7.8.1)
- if the `EndOfSequence` parameter is TRUE, the `next` attribute is omitted from the `InstanceCollection` payload element in the HTTP response (see 7.8.1)

If the `$refer` or `$expand` query parameters are specified, this CIM-RS operation maps to the generic operation `OpenClassInstancesWithPath` as described above, and possibly additional association traversal operations, as described in B.2.1.

Restrictions:

- Including the class origin of properties in the returned instance representations is not supported in CIM-RS.
- Excluding subclass properties in the returned instance representations by setting a single indicator is not supported in CIM-RS (they can be excluded through the `$properties` query parameter).

B.3.9 GET server entry point resource

This CIM-RS operation retrieves the server entry point resource (see 7.12.2), which describes optional capabilities of the CIM-RS support, and information about the CIM namespaces of the server.

This CIM-RS operation does not map to any generic operation.

The CIM namespaces can be determined through the generic operation `GetInstance` on class `CIM_Namespace` in the Interop namespace. Alternatively, this information can be retrieved through direct interfaces.

Restrictions: None.

B.3.10 PUT instance resource

This CIM-RS operation modifies some or all property values of an instance resource (see 7.6.4).

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

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

- the `InstancePath` parameter is formed from CIM namespace, class name and key bindings, which are all formed from information units extracted from the target URI of the HTTP request (see B.1.2)
- the `ModifiedInstance` parameter is formed from the `instance` attribute of the `Instance` payload element in the HTTP request (see 7.6.1)
- the `IncludedProperties` parameter is obtained from the `$properties` query parameter as described in Table B-1

This generic operation does not have any output parameters.

Restrictions: None.

B.4 Listener operations

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

The following subclauses describe each combination of HTTP method and resource type.

B.4.1 POST listener destination resource

This CIM-RS listener operation delivers an indication to a listener (see 7.11.2).

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

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

- the `ListenerDestination` parameter is formed from information units extracted from the target URI of the HTTP request (see B.1.8)
- the `Indication` parameter is formed from the `indication` attribute of the `IndicationDeliveryRequest` payload element in the HTTP request (see 7.11.1)

This generic operation does not have any output parameters.

Restrictions: None.

B.4.2 GET listener entry point resource

This CIM-RS operation retrieves the listener entry point resource (see 7.13.2), which describes optional capabilities of the CIM-RS support.

This CIM-RS operation does not map to any generic operation.

Restrictions: None.

ANNEX C
(informative)

Mapping generic operations to CIM-RS

This annex describes how generic operations (see DSP0223) are to be mapped to CIM-RS operations, resources, and query parameters. This mapping is provided primarily to describe how the CIM-RS protocol conforms to generic operations. This mapping can also be used to translate operation requirements defined in management profiles that are stated in terms of generic operations, into CIM-RS operations. The latter may be useful for implementations of CIM servers that define their provider API in terms of 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 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 client side control of returning class origin information (generic operation parameter `IncludeClassOrigin`)
- 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 enumeration operations. However, only the upcoming DMTF *Filter Query Language* will be supported. In anticipation of that, the `FilterQueryLanguage` parameter of any generic operations is set to `"DMTF:FQL"`..
- CIM-RS supports client side control of continuation on error for pulled instance enumeration operations

C.3 Operations supported

This subclause describes generic operations that are supported in CIM-RS.

C.3.1 GetInstance

This generic operation is supported via HTTP GET on an instance resource (see 7.6.3).

Its input parameters map to CIM-RS as follows:

- `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- `IncludeClassOrigin`: Not supported in CIM-RS (optional in DSP0223)
- `IncludedProperties`: `$properties` query parameter (see Table B-1)

Its output parameters map to CIM-RS as follows:

- `Instance`: `Instance` payload element in HTTP response (see 7.6.1)

Conformance: Yes.

C.3.2 DeleteInstance

This generic operation is supported via HTTP DELETE on an instance resource (see 0).

Its input parameters map to CIM-RS as follows:

- `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)

This generic operation has no output parameters.

Conformance: Yes.

C.3.3 ModifyInstance

This generic operation is supported via HTTP PUT on an instance resource (see 7.6.4).

Its input parameters map to CIM-RS as follows:

- `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- `ModifiedInstance`: `Instance` payload element in HTTP request (see 7.6.1)
- `IncludedProperties`: `$properties` query parameter (see Table B-1)

This generic operation has no output parameters.

Conformance: Yes.

C.3.4 CreateInstance

This generic operation is supported via HTTP POST on an instance creation resource (see 7.5.1).

Its input parameters map to CIM-RS as follows:

- `ClassPath`: Information units in target URI of the HTTP request (see B.1.1)
- `NewInstance`: `Instance` payload element in HTTP request (see 7.6.1)

Its output parameters map to CIM-RS as follows:

- `InstancePath`: `Location` header field in HTTP response (see 7.5.1)

Conformance: Yes.

C.3.5 OpenClassInstancesWithPath

This generic operation is supported via HTTP GET on an instance enumeration resource (see 7.9.1).

Its input parameters map to CIM-RS as follows:

- `EnumClassPath`: Information units in target URI of the HTTP request (see B.1.5)
- `FilterQueryString`: `$filter` query parameter (see Table B-1)
- `FilterQueryLanguage`: Only `"DMTF:FQL"` is supported by CIM-RS (see C.2)
- `IncludeClassOrigin`: Not supported in CIM-RS (optional in DSP0223)
- `IncludedProperties`: `$properties` query parameter (see Table B-1)
- `ExcludeSubclassProperties`: Not supported directly; can be achieved with `$properties` query parameter (see Table B-1)

- `OperationTimeout`: `$pagingtimeout` query parameter (see Table B-1)
- `ContinueOnError`: `$continueonerror` query parameter (see Table B-1)
- `MaxObjectCount`: `$max` query parameter (see Table B-1)

Its output parameters map to CIM-RS as follows:

- `InstanceList`: `instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)

Conformance: Yes.

C.3.6 OpenClassInstancePaths

This generic operation is supported via HTTP GET on an instance enumeration resource (see 7.9.1), where its `$properties` query parameter is set to include no properties.

Its input parameters map to CIM-RS as follows:

- `EnumClassPath`: Information units in target URI of the HTTP request (see B.1.5)
- `FilterQueryString`: `$filter` query parameter (see Table B-1)
- `FilterQueryLanguage`: Only `"DMTF:FQL"` is supported by CIM-RS (see C.2)
- `OperationTimeout`: `$pagingtimeout` query parameter (see Table B-1)
- `ContinueOnError`: `$continueonerror` query parameter (see Table B-1)
- `MaxObjectCount`: `$max` query parameter (see Table B-1)

Its output parameters map to CIM-RS as follows:

- `InstancePathList`: `instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)

Conformance: Yes.

C.3.7 OpenAssociatedInstancesWithPath

This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a `$properties` query parameter that specifies not to include any properties, and with a `$expand` query parameter that specifies each association to be traversed (for example, `$expand=AssociationClassName.[AssociatedClassName]AssociatedRoleName`).

Its input parameters map to CIM-RS as follows:

- `SourceInstancePath`: Information units in target URI of the HTTP request (see B.1.2)

- `AssociationClassName`: association class in `$expand` query parameter (see B.2.1)
- `AssociatedClassName`: associated class filter in `$expand` query parameter (see B.2.1)
- `SourceRoleName`: Not supported in CIM-RS (mandatory in DSP0223)
- `AssociatedRoleName`: association end in `$expand` query parameter (see Table B-1)
- `FilterQueryString`: `$filter` query parameter (see Table B-1)
- `FilterQueryLanguage`: Only `"DMTF:FQL"` is supported by CIM-RS (see C.2)
- `IncludeClassOrigin`: Not supported in CIM-RS (optional in DSP0223)
- `IncludedProperties`: `$properties` query parameter (see Table B-1) specifying properties in the navigation properties included via the `$expand` query parameter
- `ExcludeSubclassProperties`: Not supported directly; can be achieved with the `$properties` query parameter (see Table B-1) specifying properties in the navigation properties included via the `$expand` query parameter
- `OperationTimeout`: `$pagingtimeout` query parameter (see Table B-1)
- `ContinueOnError`: `$continueonerror` query parameter (see Table B-1)
- `MaxObjectCount`: `$max` query parameter (see Table B-1)

Its output parameters map to CIM-RS as follows:

- `InstanceList`: `instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)

Conformance: No, for the following reasons:

- the mandatory `SourceRoleName` filter is not supported
- traversal of all referencing associations without knowing them upfront is not supported

C.3.8 OpenAssociatedInstancePaths

This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a `$properties` query parameter that specifies not to include any properties, and with a `$refer` query parameter that specifies each association to be traversed (for example, `$refer=AssociationClassName.[AssociatedClassName]AssociatedRoleName`).

Its input parameters map to CIM-RS as follows:

- `SourceInstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- `AssociationClassName`: association class in `$refer` query parameter (see B.2.1)
- `AssociatedClassName`: associated class filter in `$refer` query parameter (see B.2.1)
- `SourceRoleName`: Not supported in CIM-RS (mandatory in DSP0223)
- `AssociatedRoleName`: association end in `$refer` query parameter (see B.2.1)

- `FilterQueryString`: `$filter` query parameter (see Table B-1)
- `FilterQueryLanguage`: Only `"DMTF:FQL"` is supported by CIM-RS (see C.2)
- `IncludeClassOrigin`: Not supported in CIM-RS (optional in DSP0223)
- `IncludedProperties`: `$properties` query parameter (see Table B-1) specifying properties in the navigation properties included via the `$refer` query parameter
- `ExcludeSubclassProperties`: Not supported directly; can be achieved with the `$properties` query parameter (see Table B-1) specifying properties in the navigation properties included via the `$refer` query parameter
- `OperationTimeout`: `$pagingtimeout` query parameter (see Table B-1)
- `ContinueOnError`: `$continueonerror` query parameter (see Table B-1)
- `MaxObjectCount`: `$max` query parameter (see Table B-1)

Its output parameters map to CIM-RS as follows:

- `InstancePathList`: `instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)

Conformance: No, for the following reasons:

- the mandatory `SourceRoleName` filter is not supported
- traversal of all referencing associations without knowing them upfront is not supported

C.3.9 OpenReferencingInstancesWithPath

This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a `$properties` query parameter that specifies not to include any properties, and with a `$expand` query parameter that specifies each association to be returned (for example, `$expand=AssociationClassName`).

Its input parameters map to CIM-RS as follows:

- `SourceInstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- `AssociationClassName`: association class in `$expand` query parameter (see B.2.1)
- `AssociatedClassName`: associated class filter in `$expand` query parameter (see B.2.1)
- `SourceRoleName`: Not supported in CIM-RS (mandatory in DSP0223)
- `AssociatedRoleName`: association end in `$expand` query parameter (see B.2.1)
- `FilterQueryString`: `$filter` query parameter (see Table B-1)
- `FilterQueryLanguage`: Only `"DMTF:FQL"` is supported by CIM-RS (see C.2)
- `IncludeClassOrigin`: Not supported in CIM-RS (optional in DSP0223)
- `IncludedProperties`: `$properties` query parameter (see Table B-1) specifying properties in the navigation properties included via the `$expand` query parameter

- ExcludeSubclassProperties: Not supported directly; can be achieved with the $properties query parameter (see Table B-1) specifying properties in the navigation properties included via the $expand query parameter
- OperationTimeout: $pagingtimeout query parameter (see Table B-1)
- ContinueOnError: $continueonerror query parameter (see Table B-1)
- MaxObjectCount: $max query parameter (see Table B-1)

Its output parameters map to CIM-RS as follows:

- InstanceList: instances attribute of InstanceCollection payload element in HTTP response (see 7.8.1)
- EnumerationContext: information units in URI of next attribute of InstanceCollection payload element in HTTP response (see 7.8.1)
- EndOfSequence: omission or presence of next attribute of InstanceCollection payload element in HTTP response (see 7.8.1)

Conformance: No, for the following reasons:

- the mandatory SourceRoleName filter is not supported
- return of all referencing associations without knowing them upfront is not supported

C.3.10 OpenReferencingInstancePaths

This generic operation is supported via HTTP GET on an instance resource (see 7.6.3), with a $properties query parameter that specifies not to include any properties, and with a $refer query parameter that specifies each association to be returned (for example, $refer=AssociationClassName).

Its input parameters map to CIM-RS as follows:

- SourceInstancePath: Information units in target URI of the HTTP request (see B.1.2)
- AssociationClassName: association class in $refer query parameter (see B.2.1)
- AssociatedClassName: associated class filter in $refer query parameter (see B.2.1)
- SourceRoleName: Not supported in CIM-RS (mandatory in DSP0223)
- AssociatedRoleName: association end in $refer query parameter (see B.2.1)
- FilterQueryString: $filter query parameter (see Table B-1)
- FilterQueryLanguage: Only "DMTF:FQL" is supported by CIM-RS (see C.2)
- IncludeClassOrigin: Not supported in CIM-RS (optional in DSP0223)
- IncludedProperties: $properties query parameter (see Table B-1) specifying properties in the navigation properties included via the $refer query parameter
- ExcludeSubclassProperties: Not supported directly; can be achieved with the $properties query parameter (see Table B-1) specifying properties in the navigation properties included via the $refer query parameter
- OperationTimeout: $pagingtimeout query parameter (see Table B-1)
- ContinueOnError: $continueonerror query parameter (see Table B-1)

- `MaxObjectCount`: `$max` query parameter (see Table B-1)

Its output parameters map to CIM-RS as follows:

- `InstancePathList`: `instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)

Conformance: No, for the following reasons:

- the mandatory `SourceRoleName` filter is not supported
- return of all referencing associations without knowing them upfront is not supported

C.3.11 PullInstancesWithPath

This generic operation is supported via HTTP GET on a page of an instance collection resource (see 7.8.2), that had been created (via the `$properties` query parameter) such that properties were to be returned.

Its input parameters map to CIM-RS as follows:

- `NamespacePath`: Information units in target URI of the HTTP request (see B.1.2)
- `EnumerationContext`: information units in target URI of the HTTP request (see B.1.2)
- `MaxObjectCount`: `$max` query parameter (see Table B-1)

Its output parameters map to CIM-RS as follows:

- `InstanceList`: `instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)

Conformance: Yes.

C.3.12 PullInstancePaths

This generic operation is supported via HTTP GET on a page of an instance collection resource (see 7.8.2), that had been created (via the `$properties` query parameter) such that no properties were to be returned.

Its input parameters map to CIM-RS as follows:

- `NamespacePath`: Information units in target URI of the HTTP request (see B.1.2)
- `EnumerationContext`: information units in target URI of the HTTP request (see B.1.2)
- `MaxObjectCount`: `$max` query parameter (see Table B-1)

Its output parameters map to CIM-RS as follows:

- `InstanceList`: `instances` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EnumerationContext`: information units in URI of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)
- `EndOfSequence`: omission or presence of `next` attribute of `InstanceCollection` payload element in HTTP response (see 7.8.1)

Conformance: Yes.

C.3.13 InvokeMethod

This generic operation is supported via HTTP POST on a non-static method invocation resource (see 7.10.3).

Its input parameters map to CIM-RS as follows:

- `InstancePath`: Information units in target URI of the HTTP request (see B.1.2)
- `MethodName`: `method` attribute of `MethodRequest` payload element in HTTP request (see 7.10.1)
- `InParmValues`: `parameters` attribute of `MethodRequest` payload element in HTTP request (see 7.10.1)

Its output parameters map to CIM-RS as follows:

- `OutParmValues`: `parameters` attribute of `MethodResponse` payload element in HTTP response (see 7.10.2)
- `ReturnValue`: `returnvalue` attribute of `MethodResponse` payload element in HTTP response (see 7.10.2)

Conformance: Yes.

C.3.14 InvokeStaticMethod

This generic operation is supported via HTTP POST on a static method invocation resource (see 7.10.3).

Its input parameters map to CIM-RS as follows:

- `ClassPath`: Information units in target URI of the HTTP request (see B.1.2)
- `MethodName`: `method` attribute of `MethodRequest` payload element in HTTP request (see 7.10.1)
- `InParmValues`: `parameters` attribute of `MethodRequest` payload element in HTTP request (see 7.10.1)

Its output parameters map to CIM-RS as follows:

- `OutParmValues`: `parameters` attribute of `MethodResponse` payload element in HTTP response (see 7.10.2)
- `ReturnValue`: `returnvalue` attribute of `MethodResponse` payload element in HTTP response (see 7.10.2)

Conformance: Yes.

C.4 Operations not supported

The following generic operations are not supported in CIM-RS.

C.4.1 Direct instance enumeration operations

Direct instance enumeration operations are not supported in CIM-RS, because it is always possible that the resulting collections in CIM-RS are paged.

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

Unsupported Direct Enumeration Operation	Supported Pulled Equivalent
GetClassInstancesWithPath	OpenClassInstancesWithPath (Section C.3.5)
GetClassInstancePaths	OpenClassInstancePaths (Section C.3.6)
GetAssociatedInstancesWithPath	OpenAssociatedInstancesWithPath (Section C.3.7)
GetAssociatedInstancePaths	OpenAssociatedInstancePaths (Section C.3.8)
GetReferencingInstancesWithPath	OpenReferencingInstancesWithPath (Section C.3.9)
GetReferencingInstancesPaths	OpenReferencingInstancePaths (Section C.3.10)

C.4.2 Class and qualifier type operations

Class and qualifier type operations are not supported in CIM-RS.

- GetClass
- DeleteClass
- ModifyClass
- CreateClass
- GetTopClassesWithPath
- GetTopClassPaths
- GetSubClassesWithPath
- GetSubClassPaths
- GetAssociatedClassesWithPath
- GetAssociatedClassPaths
- GetReferencingClassesWithPath
- GetReferencingInstancePaths
- GetQualifierType
- DeleteQualifierType
- CreateQualifierType
- EnumerateQualifierTypesWithPath

C.4.3 Other operations

The following other generic operations are not supported in CIM-RS.

- OpenQueryInstances
- PullInstances
- EnumerationCount
- CloseEnumeration

ANNEX D
(informative)

Examples

D.1 Navigation between resources

This annex provides examples on how to navigate between resources using the $expand (see 6.5.3) and $refer (see 6.5.9) query parameters. For a description of the concepts for navigating between resources, see 5.6.

D.1.1 Classes and instances used in the examples

The examples use the classes from the class diagram shown in Figure D-1.

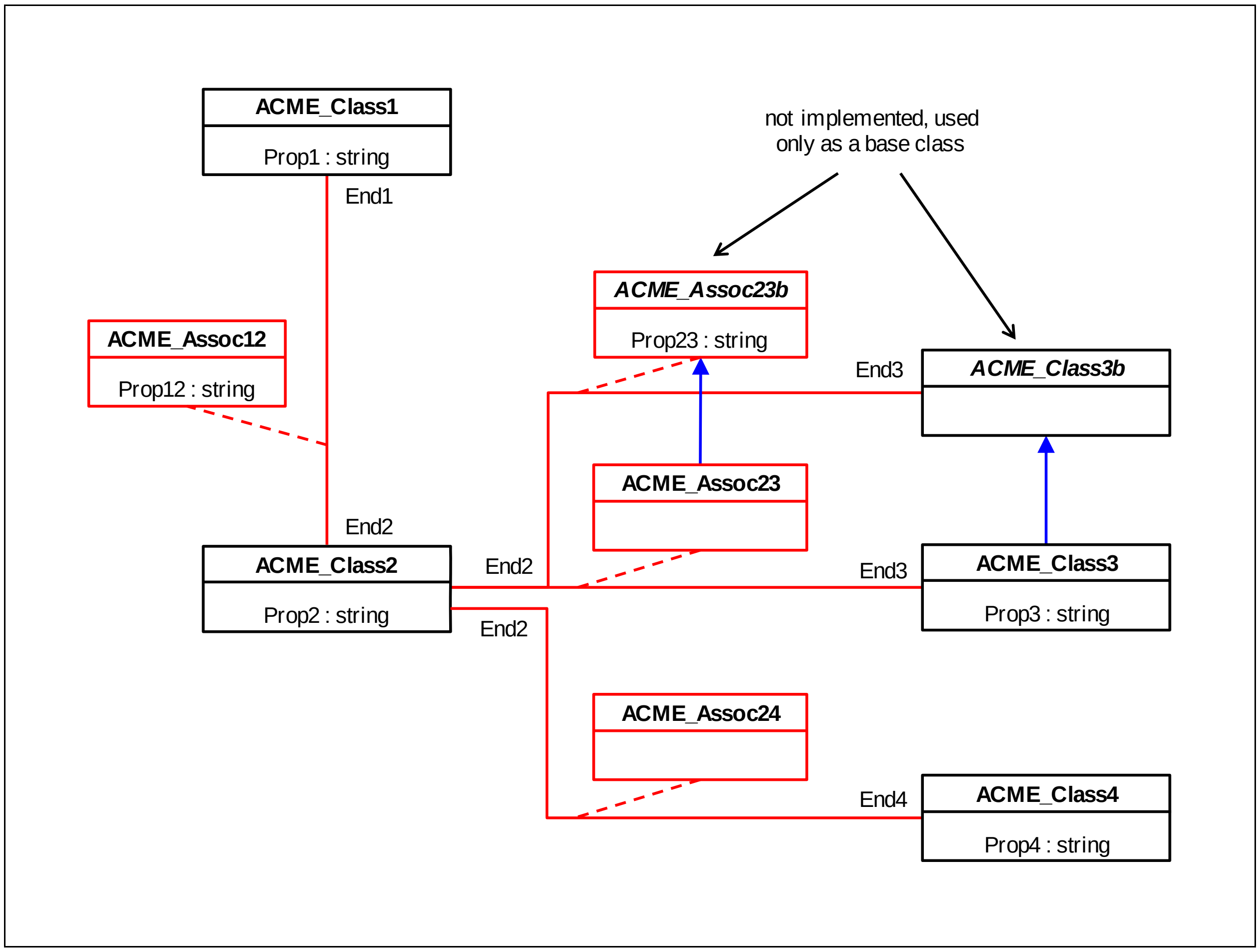

Figure D-1 – Class diagram for navigation examples

The representations of results uses an informal notation that indicates nesting of elements by indentation.

The examples are limited to requests for instance retrieval, for brevity. Requests for retrieval of instance collections work the same way, except that each instance in the collection is affected.

The following MOF defines the classes shown in Figure D-1:

```
class ACME_Class1 { string Prop1; };

class ACME_Class2 { string Prop2; };

[Abstract]
class ACME_Class3b { };                          // not implemented

class ACME_Class3 : ACME_Class3b { string Prop3; };

[Association]
class ACME_Assoc12 {
  ACME_Class1 REF End1;
  ACME_Class2 REF End2;
  string Prop12;
};

[Association, Abstract]
class ACME_Assoc23b {                            // not implemented
  ACME_Class2 REF End2;
  ACME_Class3b REF End3;
  string Prop23;
};

[Association]
class ACME_Assoc23 : ACME_Assoc23b {
  [Override("End3")] ACME_Class3 REF End3;      // now references the subclass
};

[Association]
class ACME_Assoc24 {
  ACME_Class2 REF End2;
  ACME_Class4 REF End4;
};
```

D.1.2 Navigation to referencing association instances

In this example, the client retrieves an instance and specifies a navigation path that identifies association instances that reference the instance being retrieved. Figure D-2 shows the instance diagram and the blue navigation path "ACME_Assoc12", starting at instance c1a.

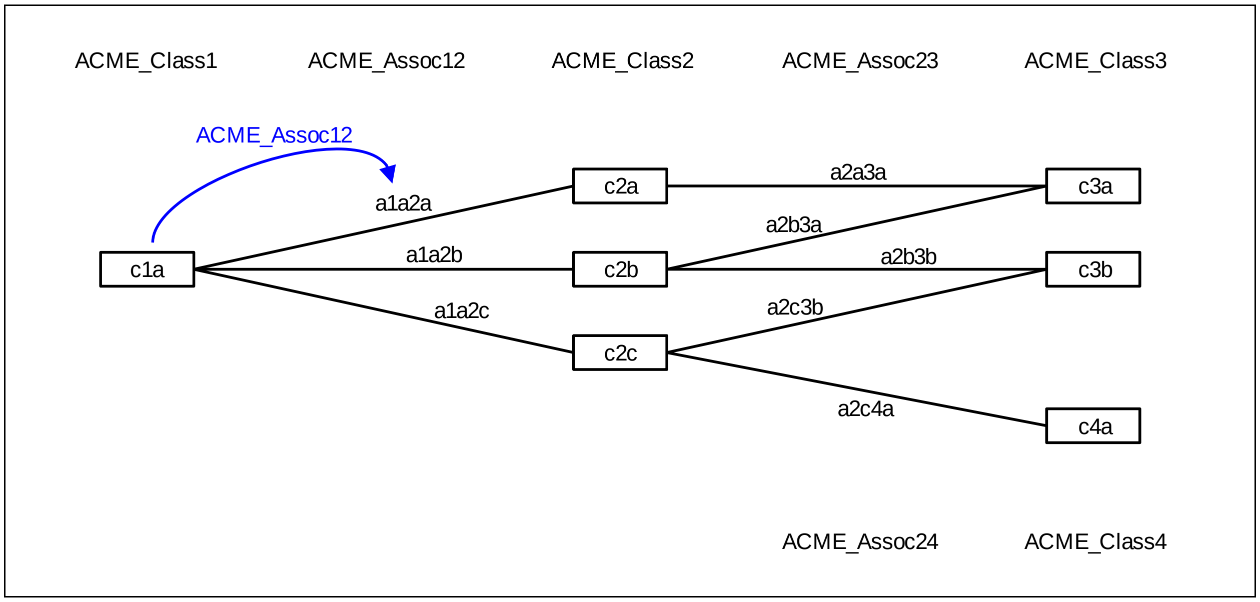

Figure D-2 – Example instance diagram for navigation to referencing association instances

An instance retrieval request using this navigation path with the $refer query parameter will return the following instance representation:

```
GET /c1a?$refer=ACME_Assoc12

Instance c1a:
   Prop1: "..."
   ACME_Assoc12: ReferenceCollection:
      ref a1a2a
      ref a1a2b
      ref a1a2c
```

An instance retrieval request using this navigation path with the $expand query parameter will return the following instance representation:

```
GET /c1a?$expand=ACME_Assoc12

Instance c1a:
   Prop1: "..."
   ACME_Assoc12: InstanceCollection:
      Instance a1a2a:
         End1: ref c1a
         End2: ref c2a
         Prop12: "..."
      Instance a1a2b:
         End1: ref c1a
         End2: ref c2b
         Prop12: "..."
      Instance a1a2c:
         End1: ref c1a
         End2: ref c2c
         Prop12: "..."
```

D.1.3 Navigation to associated instances

In this example, the client retrieves an instance and specifies a navigation path that identifies the instances associated to the instance being retrieved. Figure D-3 shows the instance diagram and the blue navigation path "ACME_Assoc12.End2", starting at instance c1a.

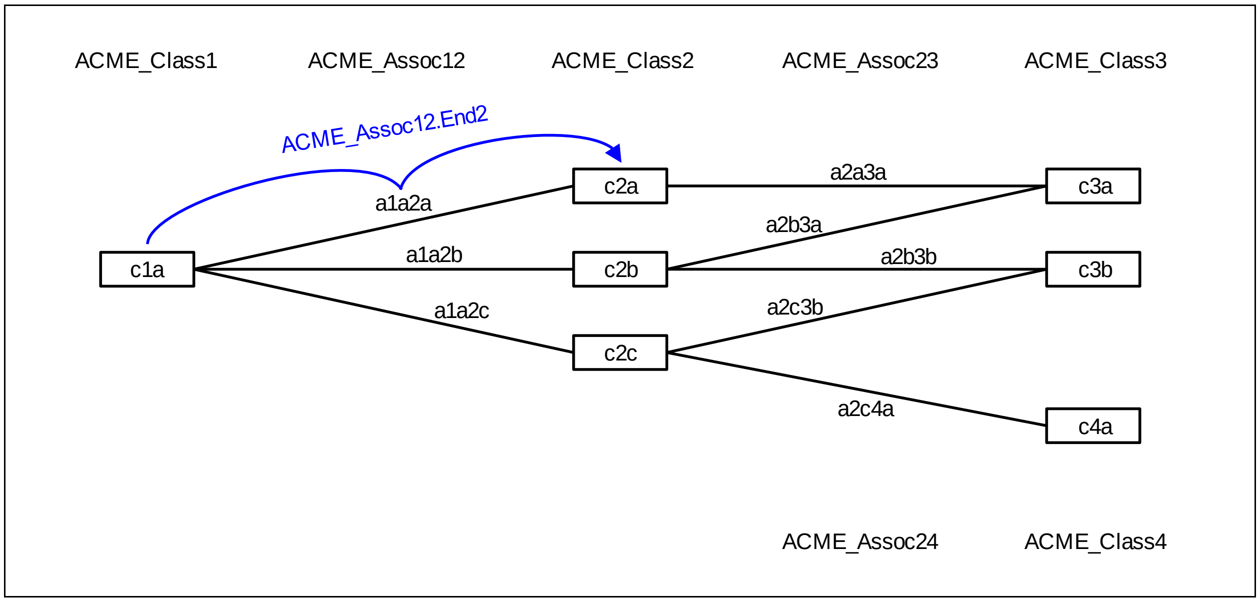

Figure D-3 – Example instance diagram for navigation to associated instances

An instance retrieval request using this navigation path with the $refer query parameter will return the following instance representation:

```
GET /c1a?$refer=ACME_Assoc12.End2

Instance c1a:
   Prop1: "..."
   ACME_Assoc12.End2: ReferenceCollection:
      ref c2a
      ref c2b
      ref c2c
```

An instance retrieval request using this navigation path with the $expand query parameter will return the following instance representation:

```
GET /c1a?$expand=ACME_Assoc12.End2

Instance c1a:
   Prop1: "..."
   ACME_Assoc12.End2: InstanceCollection:
      Instance c2a:
         Prop2: "..."
      Instance c2b:
         Prop2: "..."
      Instance c2c:
         Prop2: "..."
```

D.1.4 Navigation to association instances across one hop

In this example, the client retrieves an instance and specifies a navigation path that identifies the association instances that reference the instances associated to the instance being retrieved. Figure D-4 shows the instance diagram and the blue navigation path "ACME_Assoc12.End2.ACME_Assoc23", starting at instance c1a.

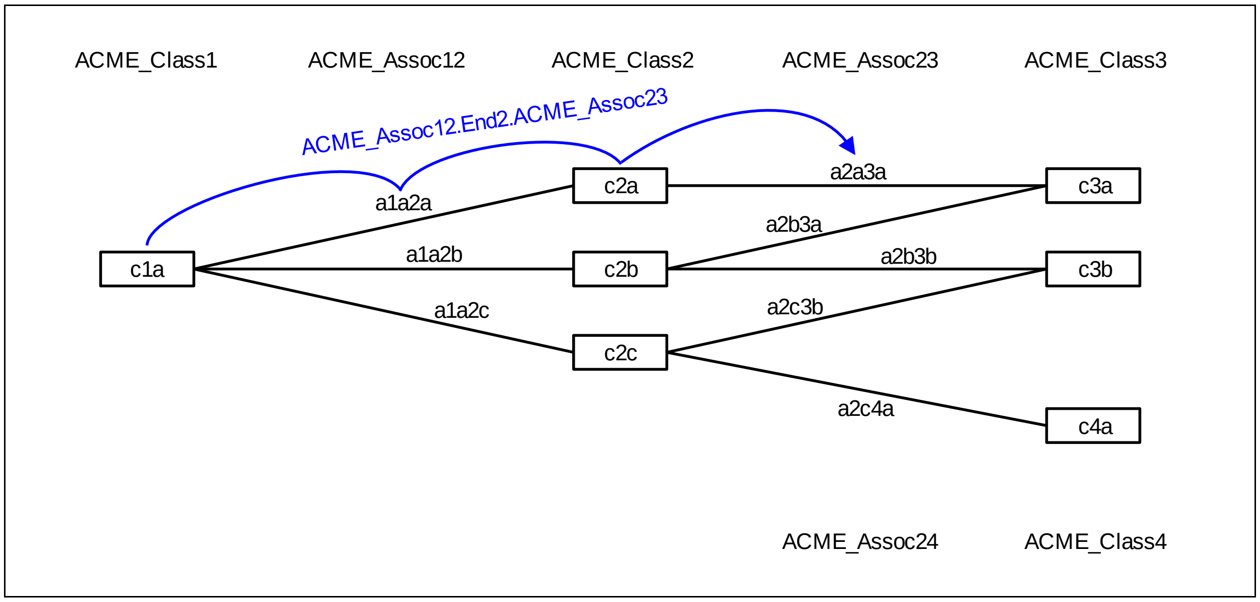

Figure D-4 – Example instance diagram for navigation to association instances across one hop

An instance retrieval request using this navigation path with the $refer query parameter will return the following instance representation:

```
GET /c1a?$refer=ACME_Assoc12.End2.ACME_Assoc23

Instance c1a:
   Prop1: "..."
   ACME_Assoc12.End2.ACME_Assoc23: ReferenceCollection:
      ref a2a3a
      ref a2b3a
      ref a2b3b
      ref a2c3b
```

Note that instances of association class ACME_Assoc24 are not included, because navigation across ACME_Assoc23 was requested.

An instance retrieval request using this navigation path with the $expand query parameter will return the following instance representation:

```
GET /c1a?$expand=ACME_Assoc12.End2.ACME_Assoc23

Instance c1a:
   Prop1: "..."
   ACME_Assoc12.End2.ACME_Assoc23: InstanceCollection:
      Instance a2a3a:
         End2: ref c2a
         End3: ref c3a
         Prop23: "..."
      Instance a2b3a:
         End2: ref c2b
         End3: ref c3a
         Prop23: "..."
      Instance a2b3b:
         End2: ref c2b
         End3: ref c3b
         Prop23: "..."
      Instance a2c3b:
         End2: ref c2c
         End3: ref c3b
         Prop23: "..."
```

D.1.5 Navigation to associated instances across two hops

In this example, the client retrieves an instance and specifies a navigation path that identifies instances associated to the instance being retrieved across two specific association hops. Figure D-5 shows the instance diagram and the blue navigation path "ACME_Assoc12.End2.ACME_Assoc23.End3", starting at instance c1a.

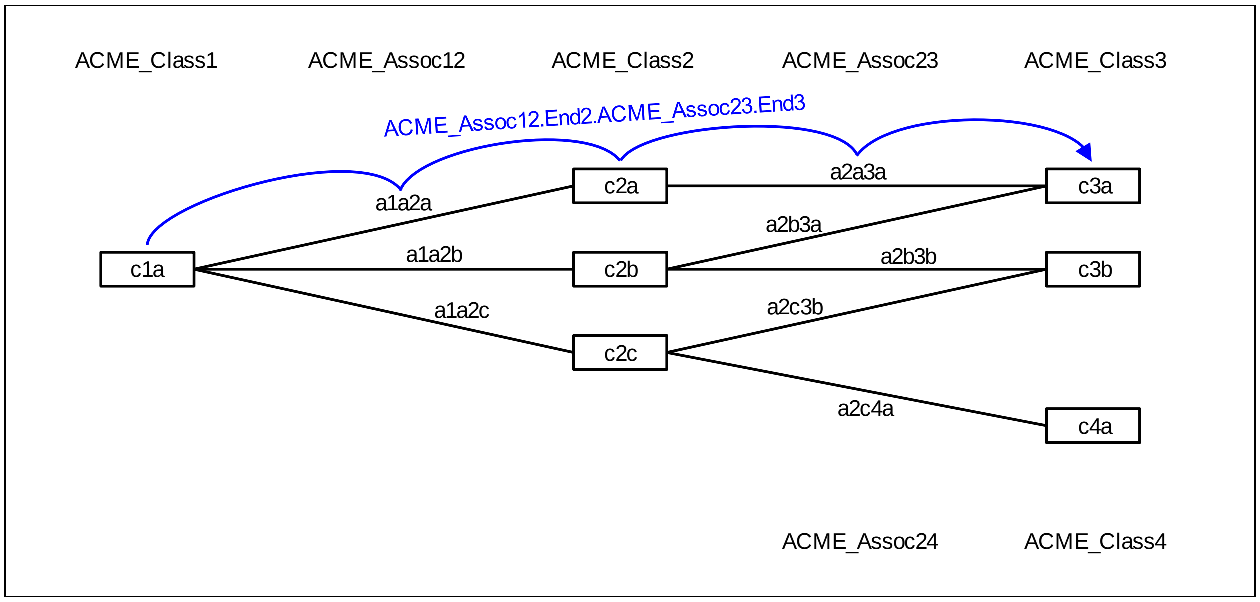

Figure D-5 – Example instance diagram for navigation to associated instances across two hops

An instance retrieval request using this navigation path with the $refer query parameter will return the following instance representation:

```
GET /c1a?$refer=ACME_Assoc12.End2.ACME_Assoc23.End3

Instance c1a:
   Prop1: "..."
   ACME_Assoc12.End2.ACME_Assoc23.End3: ReferenceCollection:
      ref c3a
      ref c3a
      ref c3b
      ref c3b
```

Note that instances c3a and c3b each occur two times in the list. The reason for this is that the inclusion is driven strictly by the navigation paths that lead to the desired target, and there is no optimization to reduce any duplicates.

Note that instances of class ACME_Class4 are not included, because navigation across ACME_Assoc23 and its End3 was requested.

An instance retrieval request using this navigation path with the $expand query parameter will also return the same duplicates and is not shown, for brevity.

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

This example is similar to the previous example, except that the navigation path uses the other possible association for the second hop. Figure D-6 shows the instance diagram and the blue navigation path "ACME_Assoc12.End2.ACME_Assoc24.End4", starting at instance c1a.

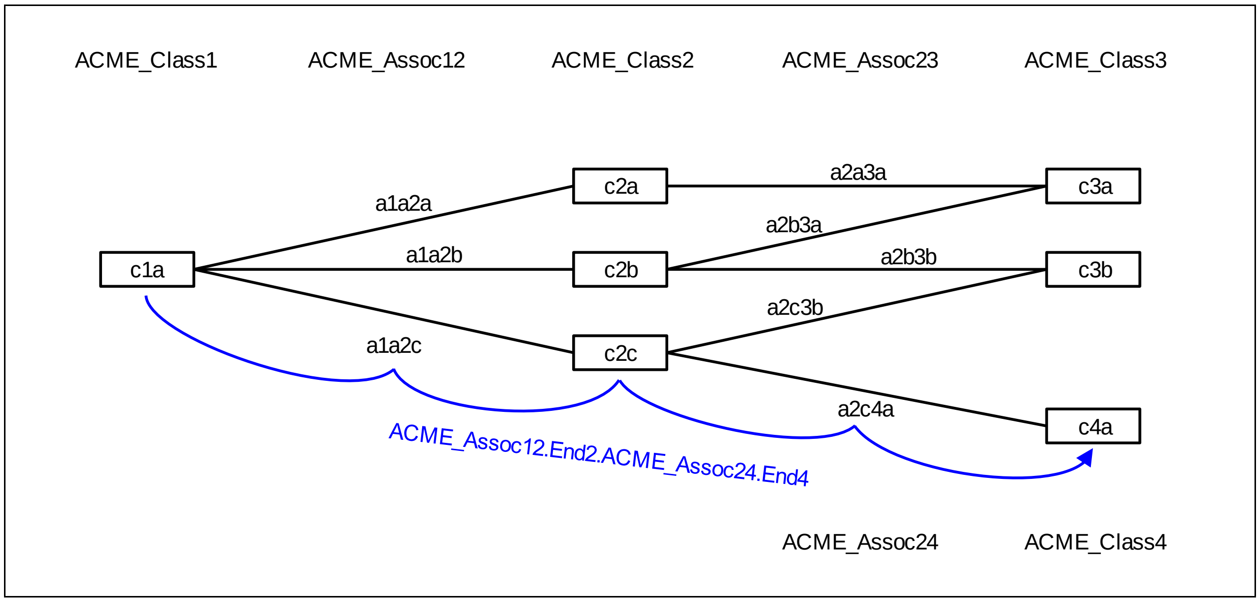

Figure D-6 – Example instance diagram for navigation to associated instances across two hops (2)

An instance retrieval request using this navigation path with the $refer query parameter will return the following instance representation:

```
GET /c1a?$refer=ACME_Assoc12.End2.ACME_Assoc24.End4

Instance c1a:
   Prop1: "..."
   ACME_Assoc12.End2.ACME_Assoc24.End4: ReferenceCollection:
      ref c4a
```

Note that the intermediate instances of class ACME_Class2 do not show up in the result. Some of them are being traversed in the course of getting to the result instances, but because only the end result is represented, the navigation path to get there does not show up.

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

In this example, the client retrieves an instance and specifies two navigation path: one that identifies instances directly associated to the instance being retrieved, and one that identifies instances associated across one additional association hop. Figure D-7 shows the instance diagram and the two navigation paths, in blue and red. The red one is a subset of the blue one, so that they can be merged if the red one is used with $expand.

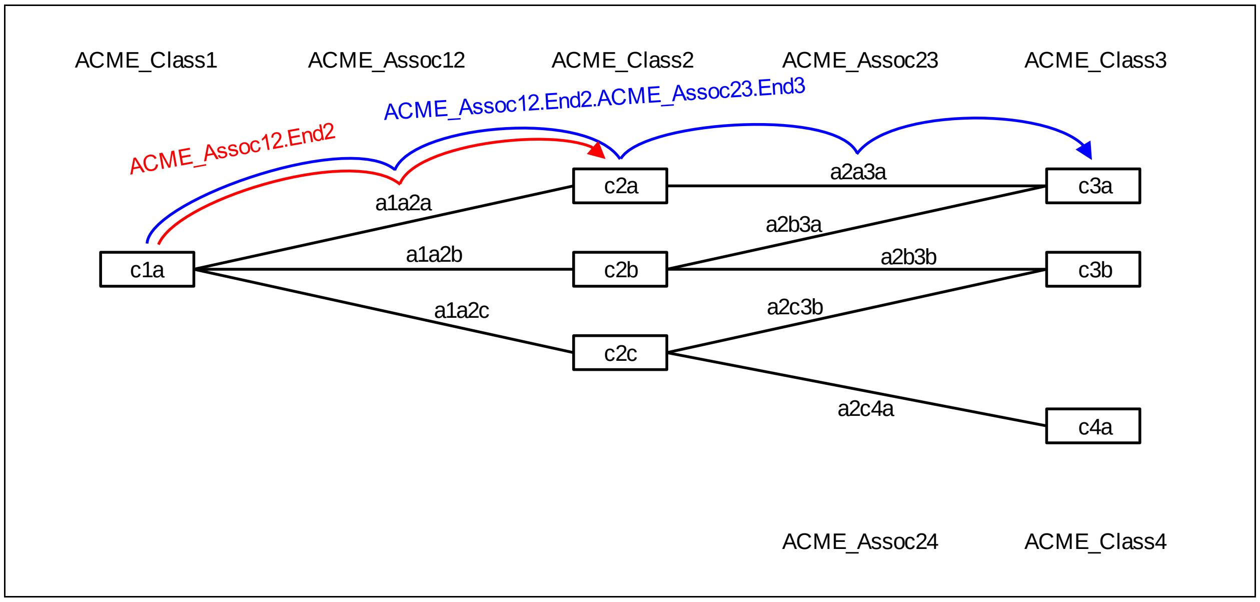

Figure D-7 – Example instance diagram for navigation with two paths that form a subset (merge)

An instance retrieval request using these two navigation paths with the $refer query parameter will return the following instance representation:

```
GET /c1a?$refer=ACME_Assoc12.End2,ACME_Assoc12.End2.ACME_Assoc23.End3

Instance c1a:
   Prop1: "..."
   ACME_Assoc12.End2: ReferenceCollection:
      ref c2a
      ref c2b
      ref c2c
   ACME_Assoc12.End2.ACME_Assoc23.End3: ReferenceCollection:
      ref c3a
      ref c3a
      ref c3b
      ref c3b
```

Note that the two navigation properties have not been merged, even though one navigation path was a subset of the other. The reason is that the shorter one was not expanded to instances.

A changed request where the shorter navigation path is used with the $expand query parameter and the longer one is used with $refer will return the following instance representation:

```
GET /c1a?$expand=ACME_Assoc12.End2&$refer=ACME_Assoc12.End2.ACME_Assoc23.End3

Instance c1a:
   Prop1: "..."
   ACME_Assoc12.End2: InstanceCollection:
       Instance c2a:
         Prop2: "..."
         ACME_Assoc23.End3: ReferenceCollection:
            ref c3a
       Instance c2b:
         Prop2: "..."
         ACME_Assoc23.End3: ReferenceCollection:
            ref c3a
            ref c3b
      Instance c2c:
         Prop2: "..."
         ACME_Assoc23.End3: ReferenceCollection:
            ref c3b
```

Note that the two navigation properties now have been merged, and that the names of the inner navigation properties are relative to their starting point (that is, just "ACME_Assoc23.End3" and not "ACME_Assoc12.End2.ACME_Assoc23.End3" as specified in the query parameter).

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

This example is similar to the previous one, except that the two navigation paths have a common path after their start but none is a subset of the other. Figure D-8 shows the instance diagram and the two navigation paths, in blue and red.

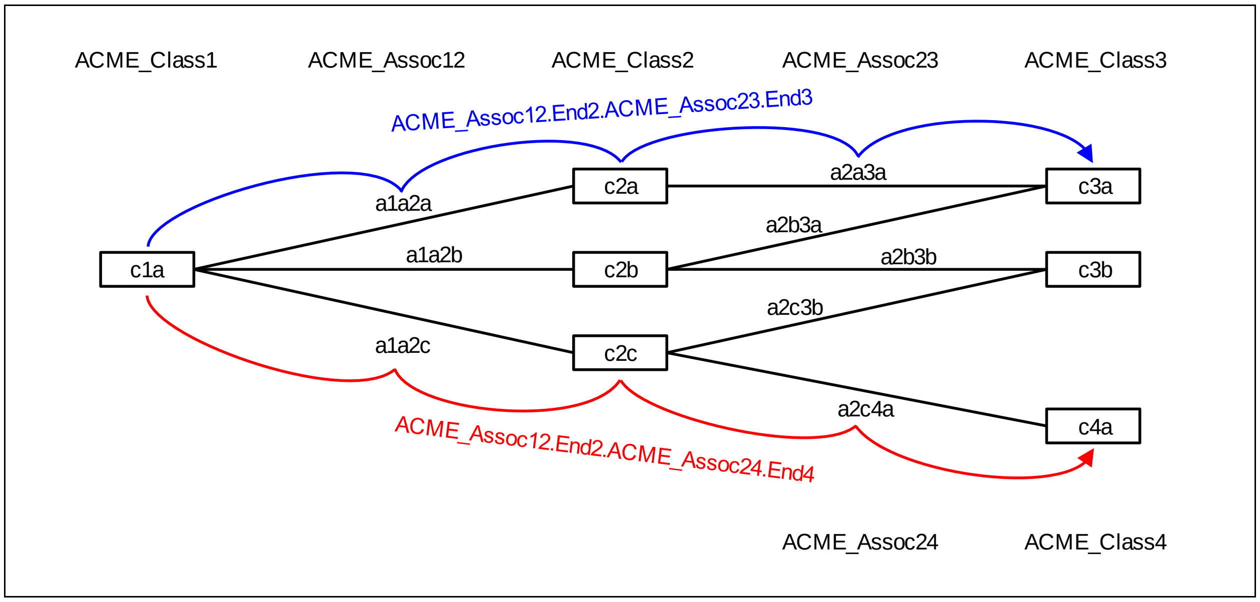

Figure D-8 – Example instance diagram for navigation with two paths that have a common begin

An instance retrieval request using these two navigation paths with the $refer query parameter will again return an instance representation with two unmerged navigation properties; it is not shown for brevity.

An instance retrieval request using one of these navigation paths with the $expand query parameter will also return an instance representation with two unmerged navigation properties:

```
GET /c1a?$expand=ACME_Assoc12.End2.ACME_Assoc23.End3&$refer=ACME_Assoc12.End2.ACME_Ass
oc24.End4

Instance c1a:
   Prop1: "..."
   ACME_Assoc12.End2.ACME_Assoc23.End3: InstanceCollection:
      Instance c3a:
         Prop3: "..."
      Instance c3a:
         Prop3: "..."
      Instance c3b:
         Prop3: "..."
      Instance c3b:
         Prop3: "..."
   ACME_Assoc12.End2.ACME_Assoc24.End4: ReferenceCollection:
        ref c4a
```

The reason for not merging is that the second property would need to have an anchor point for merging (for example, ACME_Class2 instances), and such an anchor point is not provided by the first property, because it only represents its end of the navigation path (instances referenced by End3).

This does not change even when both navigation paths are expanded, because either result is just representing the end of the navigation without providing an anchor point for the other.

D.1.9 Expansion of association reference

In this example, the client retrieves an association instance and specifies a navigation path that expands one of the existing references in the association. Figure D-9 shows the instance diagram and the blue navigation path "End2", starting at instance a1a2a.

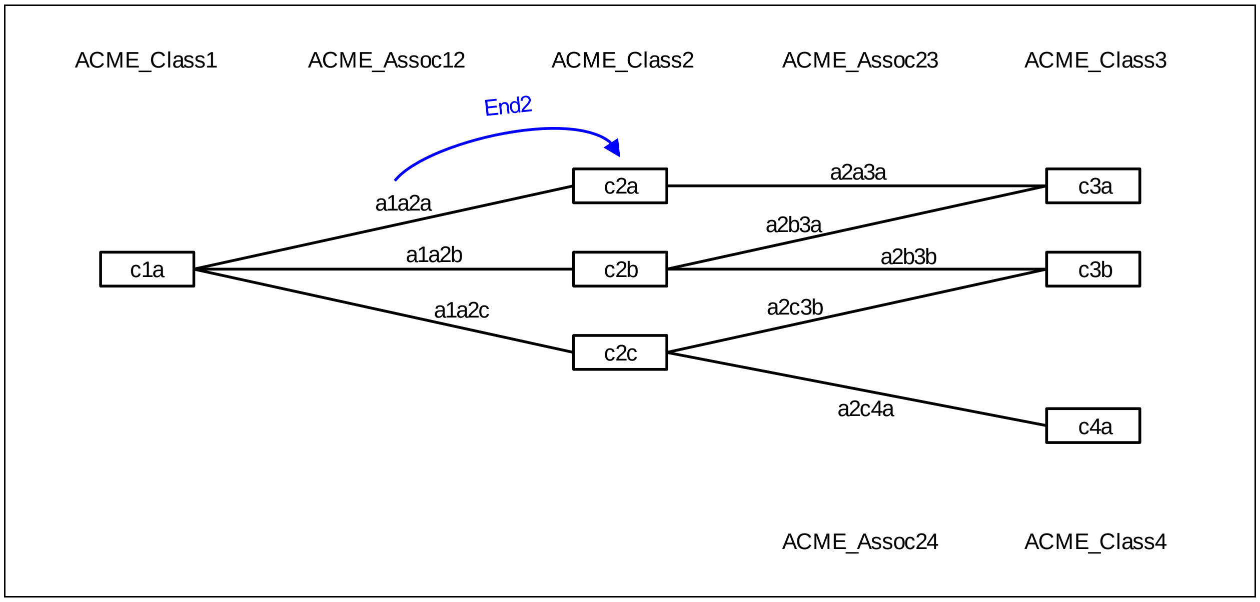

Figure D-9 – Example instance diagram for expansion of association reference

An instance retrieval request using this navigation path with the $expand query parameter will return the following instance representation:

```
GET /a1a2a?$expand=End2

Instance a1a2a:
   Prop12: "..."
   End1: ref c1a
   End2: Instance c2a:
      Prop2: "..."
```

D.1.10 Navigation from association to referencing association

In this example, the client retrieves an association instance and specifies a navigation path that identifies the association instances that reference the same instances that are also referenced by the association instance being retrieved. Figure D-10 shows the instance diagram and the blue navigation path "End2.ACME_Assoc23", starting at instance a1a2a.

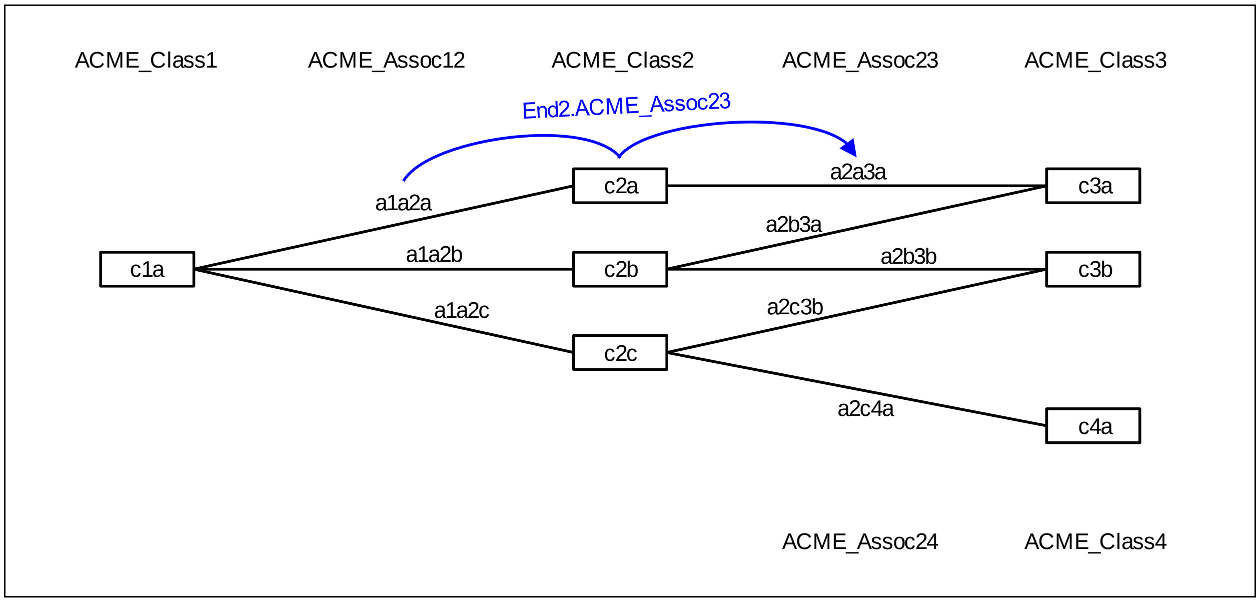

Figure D-10 – Example instance diagram for navigation starting from association

An instance retrieval request using this navigation path with the $expand query parameter will return the following instance representation:

```
GET /a1a2a?$expand=End2.ACME_Assoc12

Instance a1a2a:
   Prop12: "..."
   End1: ref c1a
   End2: ref c2a
   End2.ACME_Assoc12: InstanceCollection:
      Instance a2a3a:
         Prop23: "..."
         End2: ref c2a
         End3: ref c3a
```

D.1.11 Expansion of association reference and navigation to referencing association (merge)

In this example, the client retrieves an association instance and specifies both navigation properties from the previous two examples. Figure D-11 shows the instance diagram, the red navigation path "End2", and the blue navigation path "End2.ACME_Assoc23", both starting at instance a1a2a.

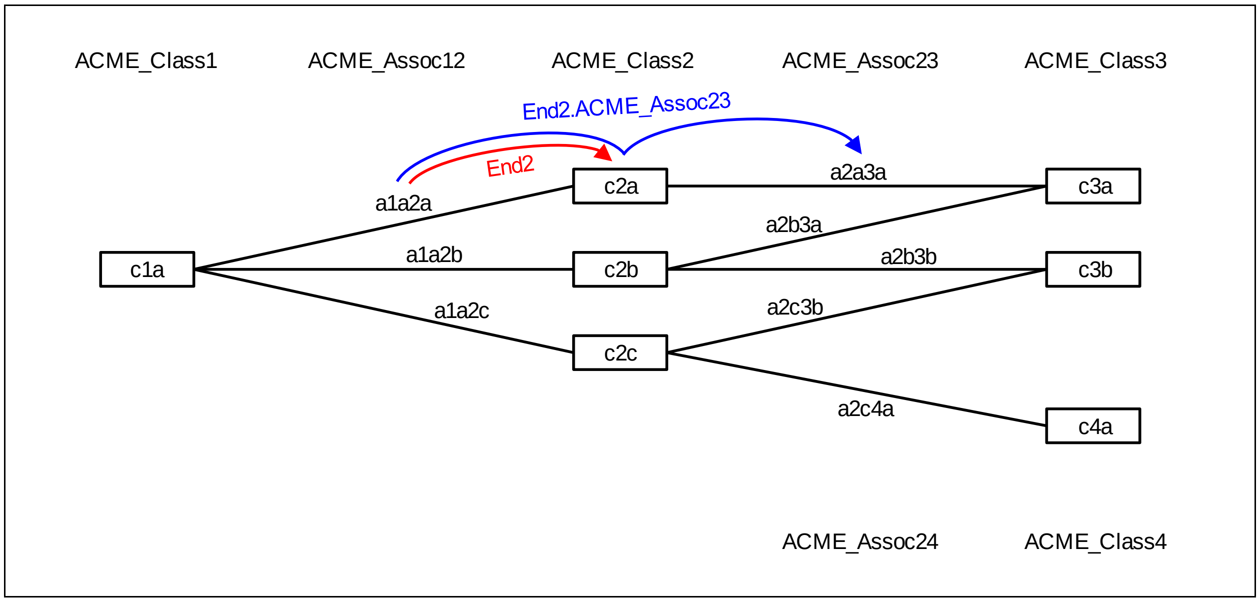

Figure D-11 – Example instance diagram for expansion of association reference and navigation to referencing association (merge)

An instance retrieval request using these navigation paths with the $expand query parameter will return the following instance representation:

```
GET /a1a2a?$expand=End2,End2.ACME_Assoc12

Instance a1a2a:
   Prop12: "..."
   End1: ref c1a
   End2: Instance c2a:                   // outer merged (existing) property
      Prop2: "..."
      ACME_Assoc12: InstanceCollection:      // inner merged navigation property
         Instance a2a3a:
            Prop23: "..."
            End2: ref c2a
            End3: ref c3a
```

The two navigation paths get merged because one is a subset of the other. The inner navigation property (specified using the navigation path "End2.ACME_Assoc12") gets merged into the existing reference "End2" and its name gets shortened to "ACME_Assoc12" because that would be the valid navigation path in the context of instance c2a.

D.2 Paged retrieval

This annex provides an example for paged retrieval, as described in 7.3.8. The example is based on the classes defined in D.1 and assumes that the client has specified a maximum size for pageable collections of 2 by using the $max parameter (see 6.5.5), in order to demonstrate paging with a small number of entities.

Because the information that controls paging is represented in the payload, the requests and responses are shown in detail instead of using the abbreviated notation used in D.1.

D.2.1 Navigation to associated instances

The following exchange shows the example from D.1.3 that includes a navigation property with references to associated instances.

Request:

```
GET /cimrs/root%2Fcimv2/ACME_Class1/c1a?$refer=ACME_Assoc12.End2&$max=2 HTTP/1.1
Host: server.acme.com:5988
Accept: application/json;version=1.0
X-CIMRS-Version: 1.0.0
```

Response:

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

{
  "kind": "instance",
  "self": "/cimrs/root%2Fcimv2/ACME_Class1/c1a",
  "class": "ACME_Class1",
  "properties": {
    "Prop1": "...",
    "ACME_Assoc12.End2": {
      "kind": "referencecollection",
      "self": "/cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/1",
      "next": "/cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/2",
      "class": "ACME_Class2",
      "references": [
        "/cimrs/root%2Fcimv2/ACME_Class2/c2a",
        "/cimrs/root%2Fcimv2/ACME_Class2/c2b"
      ]
    }
  },
  "methods": { ... }
}
```

The presence of the "next" attribute in the reference collection indicates that there are more pages to retrieve, so the client issues a request to retrieve the next page of that collection:

Request:

```
GET /cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/2?$max=2
HTTP/1.1
Host: server.acme.com:5988
Accept: application/json;version=1.0
X-CIMRS-Version: 1.0.0
```

Response:

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

{
  "kind": "referencecollection",
  "self": "/cimrs/root%2Fcimv2/ACME_Class1/c1a/refer/ACME_Assoc12.End2/part/2",
  "class": "ACME_Class2",
  "references": [
    "/cimrs/root%2Fcimv2/ACME_Class2/c2c"
  ]
}
```

This time, the reference collection does not contain a next attribute, indicating that the collection is now complete.

The variant using the $expand parameter is omitted; paged retrieval works the same for that variant except that the response now contains an instance collection instead of the reference collection. See 7.8.2 for an example of an instance collection retrieval.

ANNEX E
(informative)

Change log

Version	Date	Description
1.0.0	2013-01-24	

Bibliography

This annex contains a list of non-normative 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 1.0*, http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf

DMTF DSP1054, *Indications Profile 1.2*, http://www.dmtf.org/sites/default/files/standards/documents/DSP1054_1.2.pdf

DMTF DSP2032, *CIM-RS White Paper 1.0*, http://www.dmtf.org/standards/published_documents/DSP2032_1.0.pdf

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

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 Draft RFC *Additional HTTP Status Codes*, Draft 04, February 2012, http://tools.ietf.org/html/draft-nottingham-http-new-status-04

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/

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/