Document Number: DSP0200

Date: 2013-05-07

Version: 1.4.0a

CIM Operations over HTTP

Information for Work-in-Progress version:

IMPORTANT: This document is not a standard. It does not necessarily reflect the views of the DMTF or all of its members. Because this document is a Work in Progress, it may still change, perhaps profoundly. This document is available for public review and comment until the stated expiration date.

It expires on: 2013-10-31

Provide any comments through the DMTF Feedback Portal:
http://www.dmtf.org/standards/feedback

Document Type: Specification

Document Status: Work in Progress

Document Language: en-US

CONTENTS

Tables

Foreword

CIM Operations over HTTP (DSP0200) was prepared by the DMTF CIM-XML Working Group.

Introduction

This document defines a mapping of CIM-XML messages to the Hypertext Transfer Protocol (HTTP and HTTPS) so that implementations of CIM can operate in an open, standardized manner. It also defines the notion of *conformance* in the context of this mapping, and it describes the behavior an implementation of CIM shall exhibit to be a conforming CIM implementation.

Unless otherwise noted, the term HTTP is used in this document to mean both HTTP and HTTPS.

This document is structured as follows:

- Clause 5 describes the CIM-XML messages that form the HTTP payload using XML. It specifies the syntax and semantics of the message requests and their corresponding responses.
- Clause 6 describes the encapsulation of these messages in HTTP request and response messages, with examples of each. It also describes the extension headers used to convey additional CIM-specific semantics in the HTTP Header.
- Clause 7 presents details of other aspects of the encapsulation:
  - HTTP version support
  - Use of standard HTTP headers
  - HTTP error codes
  - Security considerations

Requirements

There are many different ways CIM-XML messages can be represented in XML and encapsulated within HTTP messages. To attain interoperability among different implementations of CIM, both the XML representation and the HTTP encapsulation must be standardized. The XML representation is defined in DSP0201. DSP0203 and DSP8044 define the DTD and XSD for that XML representation, for convenience. This document uses that XML representation to define the HTTP encapsulation.

The following criteria are applied to the representation of CIM-XML messages in XML using DSP0201:

- Each CIM-XML message is completely described in XML; completeness is favored over conciseness.
- The set of CIM-XML messages provides enough functionality to enable implementations of CIM to communicate effectively for management purposes. This release of the mapping does not provide a *complete* set of messages. Rather, the goal is to define the mapping so that it admits straightforward extension (by the addition of further features) in future versions.
- The set of CIM-XML messages is classified into functional profiles to accommodate a range of implementations varying from complete support of all messages to support of a minimal subset. The number of functional profiles is kept as small as possible to encourage interoperability, and mechanisms provided by different CIM implementations can declare their level of support.

The following criteria are applied to the HTTP encapsulation of CIM-XML messages herein:

- In recognition of the large installed base of HTTP/1.0 systems, the encapsulation is designed to support both HTTP/1.0 and HTTP/1.1. However, support for HTTP/1.0 has been deprecated in version 1.4 of this document (see 7.1).
- The encapsulation does not introduce requirements that conflict with those stated in HTTP/1.0 or HTTP/1.1.

- Use of the encapsulation should be straightforward over the current base HTTP infrastructures. Some features anticipate and exploit enhancements to this base, but no aspects of the encapsulation require such enhancements as mandatory.
- The encapsulation avoids the use of pure HTTP tunneling or URL munging (for example, the use of the "?" character) in favor of a mechanism that allows existing HTTP infrastructures to control content safely.
- The encapsulation exposes key CIM-XML message information in headers to allow efficient firewall/proxy handling. The information is limited to essentials so that it does not have a significant impact on the size of the header. All CIM-specific information in a header also appears within the CIM-XML message.
- There is a clear and unambiguous encapsulation of the CIM-XML message payload within the HTTP message. Conciseness of the encapsulation is of secondary importance.

282

CIM Operations over HTTP

1 Scope

The Common Information Model (CIM) (for details, see DSP0004) is an object-oriented information model defined by the Distributed Management Task Force (DMTF) that provides a conceptual framework for describing management data.

The Hypertext Transfer Protocol (HTTP) (RFC1945, RFC2616) is an application-level protocol for distributed, collaborative, hypermedia information systems. This generic stateless protocol can be used for many tasks through extension of its request methods, error codes, and headers.

The Hypertext Transfer Protocol Secure (HTTPS) (RFC2818) is the usage of HTTP over secure sockets provided by TLS. It supports encryption of the messages exchanged, secure identification of servers, and secure authentication of clients.

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

The Extensible Markup Language (XML) is a simplified subset of SGML that offers powerful and extensible data modeling capabilities. An *XML document* is a collection of data represented in XML. An *XML schema* is a grammar that describes the structure of an XML document.

This document defines a mapping of CIM-XML messages onto HTTP that allows implementations of CIM to interoperate in an open, standardized manner. It is based on DSP0201 that defines the XML schema for CIM objects and messages.

2 Normative References

The following referenced documents are indispensable for applying the information in this document while developing an implementation of CIM. For dated references, only the edition cited applies. For undated references, the latest edition applies, including any amendments.

DMTF DSP0004, *Common Information Model (CIM) Infrastructure 2.7*,
http://www.dmtf.org/standards/published_documents/DSP0004_2.7.pdf

DMTF DSP0201, *Representation of CIM in XML 2.4*,
http://www.dmtf.org/standards/published_documents/DSP0201_2.4.pdf

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

DMTF DSP0223, *Generic Operations 1.1*,
http://www.dmtf.org/standards/published_documents/DSP0223_1.1.pdf

DMTF DSP8016, *WBEM Operations Message Registry 1.1*,
http://www.dmtf.org/standards/published_documents/DSP8016_1.1.xml

IETF RFC1766, *Tags for the Identification of Languages*, March 1995,
http://www.ietf.org/rfc/rfc1766.txt

IETF RFC1945, *Hypertext Transfer Protocol – HTTP/1.0*, May 1996,
http://www.ietf.org/rfc/rfc1945.txt

IETF RFC2246, *The TLS Protocol, Version 1.0*, January 1999,
http://www.ietf.org/rfc/rfc2246.txt

IETF RFC2277, *IETF Policy on Character Sets and Languages*, January 1998, http://www.ietf.org/rfc/rfc2277.txt

IETF RFC2279, *UTF-8, a transformation format of Unicode and ISO 10646*, January 1998, http://www.ietf.org/rfc/rfc2279.txt

IETF RFC2376, *XML Media Types*, July 1998, http://www.ietf.org/rfc/rfc2376.txt

IETF RFC2396, *Uniform Resource Identifiers (URI): Generic Syntax*, August 1998, http://www.ietf.org/rfc/rfc2396.txt

IETF RFC2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999, http://www.ietf.org/rfc/rfc2616.txt

IETF RFC2617, *HTTP Authentication: Basic and Digest Access Authentication*, June 1999, http://www.ietf.org/rfc/rfc2617.txt

IETF RFC2774, *HTTP Extension Framework*, February 2000, http://www.ietf.org/rfc/rfc2774.txt

IETF RFC2818, *HTTP Over TLS*, May 2000, http://www.ietf.org/rfc/rfc2818.txt

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

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

NIST 800-57 Part 1, *Recommendation for Key Management: Part 1: General (Revision 3)*, July 2012, http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf

NIST 800-131A, *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

W3C Recommendation, *Extensible Markup Language (XML), Version 1.0*, August 2006, http://www.w3.org/TR/REC-xml-names/

W3C Recommendation, *Namespaces in XML*, January 1999, http://www.w3.org/TR/1999/REC-xml-names-19990114/

W3C, *XML Schema Part 1: Structures*, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

W3C, *XSL Transformations (XSLT), Version 1.0*, November 1999, http://www.w3.org/TR/xslt

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

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 thatISO/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 DSP0201 apply to this document. The following additional terms are used in this document. Some additional more detailed terms are defined throughout the subclauses of this document.

3.1

CIM element

one of the following components of the CIM metamodel used to define a schema: Class, instance, property, method, parameter, or qualifier

3.2

CIM object

a namespace, class, instance, or qualifier that is accessible in a WBEM server

CIM-XML protocol

the WBEM protocol that uses the CIM operations over HTTP defined in this document and the representation of CIM in XML defined in DSP0201

3.3

WBEM client

the client role in the CIM-XML protocol and in other WBEM protocols. See 6.1 for a complete definition.

3.4

WBEM listener

the event listener role in the CIM-XML protocol and in other WBEM protocols. See 6.1 for a complete definition.

3.5

WBEM protocol

a communications protocol between WBEM client, WBEM server and WBEM listener

3.6

WBEM server

the server role in the CIM-XML protocol and in other WBEM protocols. See 6.1 for a complete definition.

3.7

XML element

a component of XML that is defined using the ELEMENT construct in the DTD

4 Abbreviated Terms and Document Conventions

4.1 Abbreviated Terms

The following symbols and abbreviations are used in this document.

CIM

Common Information Model

DTD

Document Type Definition

HTTP

Hypertext Transfer Protocol

XML

Extensible Markup Language

4.2 Document Conventions

This document uses the same notational conventions and basic parsing constructs that are defined in RFC2068.

Throughout this document, any deprecated element is indicated by one of the following labels:

- The "**DEPRECATION NOTE:**" label preceding a paragraph indicates that the paragraph explains a deprecated element.
- The "**DEPRECATED.**" label before a list item indicates that the information in that list item is deprecated.
- The "**(DEPRECATED)**" label after a heading applies to the entire clause for that heading.
- The "**(DEPRECATED)**" label at the end of a line in a code fragment or an example indicates that the particular line of the code fragment or example is deprecated.

5 CIM-XML Message Syntax and Semantics

This document defines all interactions among CIM products as CIM-XML messages. A CIM-XML message is a well-defined request or response data packet for exchanging information among CIM products. The two types of CIM-XML messages are as follows:

- *CIM-XML operation message*. This type of message is used between WBEM client and WBEM server to invoke an operation on the WBEM server.
- *CIM-XML export message*. This type of message is used between WBEM server and WBEM listener to communicate information (typically an event) to a WBEM listener.

This clause describes the syntax and semantics of CIM-XML messages independently of their encapsulation within a particular protocol such as HTTP. XML is used as the basis for this description, and in particular the CIM Representation in XML (DSP0201).

Note that "CIM message" (etc.) was used for the term "CIM-XML message" (etc.) before version 1.4 of this document.

5.1 Well-Formed, Valid, and Loosely Valid Documents

In this discussion, any reference to well-formed or valid XML documents has the standard meaning defined in Extensible Markup Language (XML).

XML document type definitions (DTDs) are restricted to be either well-formed or valid. However, this document also uses the term loosely valid to apply to XML that removes any attributes or elements in the XML document that do not appear in the CIM XML DTD. The resulting document is valid with respect to the CIM XML DTD and is therefore loosely valid.

In effect, a loosely valid document is valid with respect to the CIM XML DTD apart from having additional attributes or elements not defined by that DTD. The concept is very similar to that of an open content model as defined by the working draft on XML Schemas, expressed within the more limited scope of DTDs. One corollary of this definition is that any XML document that is valid with respect to the CIM XML DTD is also loosely valid.

The motivation for introducing the loosely valid class of XML documents is to relax the restrictions on a WBEM client, WBEM server, or WBEM listener when parsing received XML documents defined within the scope of this mapping. Not all clients (including their respective WBEM servers or WBEM listeners) should be required to validate each received CIM-XML message response (or its respective CIM-XML message request) because such a requirement would place too heavy a processing burden on the validating entity at the expense of footprint and performance, most notably in communication between robust and conformant implementations of this mapping.

Instead, the following requirements are set forth in this document. In all cases, a WBEM client has a respective alternative WBEM server or WBEM listener, and a received CIM-XML message response has a respective alternative CIM-XML message request:

- A WBEM client may include a DOCTYPE element in a CIM-XML message request. If so, an external declaration should be used. In-lining of the complete DTD within a message is discouraged.
- A WBEM client may elect to validate a received CIM-XML message response.
- If a WBEM client elects not to validate a received CIM-XML message, then loose validation shall be enforced.

The behavior of a WBEM server or WBEM listener with respect to a received CIM-XML message request is covered in detail in 7.3.

5.2 Operational Semantics

The CIM Representation in XML (DSP0201) defines a child element under the root <CIM> XML element called <MESSAGE>, which contains one of the following XML child elements:

- CIM-XML operation message child elements
  - <SIMPLEREQ>
  - <SIMPLERSP>
  - <MULTIREQ>
  - <MULTIRSP>
- CIM-XML export message child elements
  - <SIMPLEXPREQ>

- <SIMPLEXPRSP>
- <MULTIEXPREQ>
- <MULTIEXPRSP>

In the remainder of this document, the following terms denote an XML document that is loosely valid with respect to the CIM XML DTD:

- *Operation request message*. Contains under the root <CIM> node a <MESSAGE> child element that has a <MULTIREQ> or <SIMPLEREQ> child element under it.
- *Operation response message*. Contains under the root <CIM> node a <MESSAGE> child element that has a <MULTIRSP> or <SIMPLERSP> child element under it.
- *Export request message*. Contains under the root <CIM> node a <MESSAGE> child element that has a <MULTIEXPREQ> or <SIMPLEEXPREQ> child element under it.
- *Export response message*. Contains under the root <CIM> node a <MESSAGE> child element that has a <MULTIEXPRSP> or <SIMPLEEXPRSP> child element under it.

The phrase "CIM-XML message request" refers to either an operation request message or an export request message. The phrase "CIM-XML message response" refers to either an operation response message or an export response message.

A CIM-XML message request shall contain a non-empty value for the ID attribute of the <MESSAGE> element. The corresponding CIM-XML message response shall supply the same value for that attribute. Clients should employ a message ID scheme that minimizes the chance of receiving a stale CIM-XML message response.

Any CIM-XML message conforming to this document shall have a minimum value of "1.0" and a maximum value that is equal to the latest version of this document for the `PROTOCOLVERSION` attribute of the <MESSAGE> element.

An operation response message sent in response to an operation request message shall specify the same value for the `ID` attribute of the <MESSAGE> element that appears in the request message and contain one of the following:

- A <MULTIRSP> child element, if the operation request message contains a <MULTIREQ> child element.
- A <SIMPLERSP> child element, if the operation request message contains a <SIMPLEREQ> child element.

A *simple operation request* is an operation request message that contains a <SIMPLEREQ> child element. A simple operation response is an Operation Response Message that contains a <SIMPLERSP> child element.

A *multiple operation request* is an operation request message that contains a <MULTIREQ> child element. A multiple operation response is an operation response message that contains a <MULTIRSP> child element.

An export response message sent in response to an export request message shall specify the same value for the ID attribute of the <MESSAGE> element that appears in the export request message and shall contain one of the following:

- A <MULTIEXPRSP> child element if the export request message contained a <MULTIEXPREQ> child element, or
- A <SIMPLEEXPRSP> child element if the export request message contained a <SIMPLEEXPREQ> child element.

A simple export request is an export request message that contains a <SIMPLEEXPREQ> child element. A simple export response is an export response message that contains a <SIMPLEEXPRSP> child element.

A multiple export request is an export request message that contains a <MULTIEXPREQ> child element. A multiple export response is an export response message that contains a <MULTIEXPRSP> child element.

5.3 Operation Correlators

5.3.1 Overview

WBEM servers may support maintaining a log to record certain aspects of operations requested by clients. The log data can provide a record of access, activity, configuration changes or audit related information. The purpose of audit related information is to identify what was done when servicing the operation, when it was done, and on behalf of which end user the operation was requested. In some environments, providing such audit information is a matter of regulatory compliance.

The credentials used for authentication with a WBEM server are not necessarily associated with the identity of an end user. For example, when the client application is a management server handling multiple end users, it is not uncommon to use the credentials of a system user (e.g. user "root" on Linux or UNIX systems) for authentication with the WBEM server. In such environments, a log on the WBEM server can only record the identity of the system user that was used for authentication, but not the identity of the end user on behalf of which the operation was requested.

Version 1.4 of this document introduced the concept of operation correlators which are named values that can be included by WBEM clients in operation request messages so that a WBEM server can add these correlators to any logs it maintains. To maintain symmetry, export request messages can also include operation correlators for use in any logs a WBEM listener may maintain.

The meaning of operation correlators is defined by the originator of the message and does not need to be understood by the receiver of the message; the receiver only stores the operation correlator along with any log entries about the message.

5.3.2 Representation

Operation correlators are represented in the CIM-XML protocol using the CORRELATOR element. Each occurence of a CORRELATOR element represents one operation correlator. For details, see DSP0201.

Zero or more operation correlators may be specified in simple operation request messages and in simple extrinsic request messages. Since the operations in a multiple operation may not have any semantic relationship within each other, the operation correlators are specified only at the level of simple operations within the multipe operation; operation correlators cannot be specified at the level of multiple operations.

This document defines no requirements on the number, content or meaning of operation correlators.

5.3.3 Implementation Requirements and Compatibility for Operation Messages

Supporting operation correlators for WBEM clients is optional. If a WBEM client implements support for operation correlators, it may include zero or more operation correlators in a simple operation request message. The number, content and meaning of operation correlators may be different in each operation.

Supporting operation correlators for WBEM servers for its operation messages is optional. If a WBEM server implements support for operation correlators for its operation messages, it shall store the operation correlators specified in a simple operation request message along with any log information about the operation. If the operation itself is not logged on the server, the correlator also does not need to be

564 logged. In order to avoid vulnerabilities by specification of excessive amounts of operation correlators,
565 WBEM servers may implement limits on operation correlators.

566 Since participants in the protocol defined by this document are required to ignore any unknown XML
567 elements in messages they receive, introducing support for operation correlators in WBEM clients is
568 compatible for WBEM servers that do not support them.

569 ### 5.3.4 Implementation Requirements and Compatibility for Export Messages

570 Supporting operation correlators for WBEM servers for its export messsages is optional. If a WBEM
571 server implements support for operation correlators for its export messsages, it may include zero or more
572 operation correlators in a simple export request message. The number, content and meaning of operation
573 correlators may be different in each export message.

574 Supporting operation correlators for WBEM listeners is optional. If a WBEM listener implements support
575 for operation correlators, it shall store the operation correlators specified in a simple export request
576 message along with any log information about the export message. If the export message itself is not
577 logged on the listener, the correlator also does not need to be logged. In order to avoid vulnerabilities by
578 specification of excessive amounts of operation correlators, WBEM listeners may implement limits on
579 operation correlators.

580 Since participants in the protocol defined by this document are required to ignore any unknown XML
581 elements in messages they receive, introducing support for operation correlators in WBEM servers for its
582 export messsages is compatible for WBEM listeners that do not support them.

583 ## 5.4 CIM Operation Syntax and Semantics

584 This clause describes method invocations, intrinsic methods, and namespace manipulation.

585 ### 5.4.1 Method Invocations

586 All CIM-XML operation requests defined for this CIM-to-HTTP mapping are defined as invocations of one
587 or more methods. A method can be either:

588 - An *intrinsic* method, which is defined for the purposes of modeling a CIM operation.
589 - An *extrinsic* method, which is defined on a CIM class in a schema.

590 In addition, intrinsic methods are made against a CIM namespace. Extrinsic methods are invoked on a
591 CIM class (if static) or instance otherwise. Intrinsic methods are defined in 5.4.2.

592 An extrinsic method call is represented in XML by the <METHODCALL> element, and the response to
593 that call is represented by the <METHODRESPONSE> element.

594 An intrinsic method call is represented in XML by the <IMETHODCALL> element, and the response to
595 that call is represented by the <IMETHODRESPONSE> element. An input parameter has an IN qualifier
596 (with a value of `true`) in the method definition, and an output parameter has an OUT qualifier (with a
597 value of `true`). A parameter can be both an input and an output parameter.

598 The <METHODCALL> or <IMETHODCALL> element names the method to be invoked and supplies any
599 input parameters to the method call. Note the following rules about parameters:

600 - Each input parameter shall be named using the name assigned in the method definition.
601 - Input parameters may be supplied in any order.
602 - Each input parameter of the method, and no others, shall be present in the call, unless it is
603 optional.

The <METHODRESPONSE> or <IMETHODRESPONSE> element defines either an <ERROR> or an optional return value and output parameters if it is decorated with the OUT qualifier in the method definition. In the latter case, the following rules about parameters apply:

- Each output parameter shall be named using the name assigned in the method definition.
- Output parameters may be supplied in any order.
- Each output parameter of the method, and no others, shall be present in the response, unless it is optional.
- The method invocation process can be thought of as the binding of the input parameter values specified as child elements of the <METHODCALL> or <IMETHODCALL> element to the input parameters of the method. This binding is followed by an attempt to execute the method using the bound input parameters with one of the following results:
  - If the attempt to call the method is successful, the return value and output parameters are bound to the child elements of the <METHODRESPONSE> or <IMETHODRESPONSE> element.
  - If the attempt to call the method is unsuccessful, an error code and optional human-readable description of that code is bound to the <METHODRESPONSE> or <IMETHODRESPONSE> element.

5.4.1.1 Simple Operations

A simple operation invokes a single method. A simple operation request is represented by a <SIMPLEREQ> element, and a simple operation response is represented by a <SIMPLERSP> element.

If the method is intrinsic, then the <SIMPLEREQ> element shall contain an <IMETHODCALL> element, which in turn contains a <LOCALNAMESPACEPATH> child element identifying the local CIM namespace against which the method is to execute. If the method is extrinsic, then the <SIMPLEREQ> element shall contain a <METHODCALL> element that in turn contains one of the following child elements:

- A <LOCALCLASSPATH> child element identifying the CIM class on which the method is to be invoked if the method is static.
- A <LOCALINSTANCEPATH> child element identifying the CIM instance on which the method is otherwise to be invoked.

5.4.1.2 Multiple Operations

A multiple operation requires the invocation of more than one method. A multiple operation request is represented by a <MULTIREQ> element, and a multiple operation response is represented by a <MULTIRSP> element.

A <MULTIREQ> (or its respective <MULTIRSP>) element is a sequence of two or more <SIMPLEREQ> (or their respective <SIMPLERSP>) elements.

A <MULTIRSP> element shall contain a <SIMPLERSP> element for every <SIMPLEREQ> element in the corresponding multiple operation response. These <SIMPLERSP> elements shall be in the same order as their <SIMPLEREQ> counterparts so that the first <SIMPLERSP> in the response corresponds to the first <SIMPLEREQ> in the request, and so forth.

Multiple operations conveniently allow multiple method invocations to be batched into a single HTTP message. Batching reduces the number of roundtrips between a WBEM client and a WBEM server and allows the WBEM server to make internal optimizations if it chooses. Note that multiple operations do not confer any transactional capabilities in processing the request. For example, the WBEM server does not have to guarantee that the constituent method calls either all fail or succeed, only that the entity make a "best effort" to process the operation. However, servers shall finish processing each operation in a

batched operation before executing the next one. Clients shall recognize that the order of operations within a batched operation is significant.

Not all WBEM servers support multiple operations; the way they declare support for this feature is defined in 7.5.

5.4.1.3 Status Codes

This clause defines the status codes and detailed error information that a conforming WBEM server application can return. The value of an <ERROR> child element within a <METHODRESPONSE> or <IMETHODRESPONSE> element includes the following parts:

- a mandatory status code
- an optional human-readable description of the status code
- zero or more CIM_Error instances

Table 1 defines the status codes that a conforming WBEM server application can return as the value of the CODE attribute of an <ERROR> child element. In addition to a status code, a conforming WBEM server may return zero or more <INSTANCE> child elements as part of an <ERROR> element. Each <INSTANCE> child element shall be an instance of CIM_Error. For each instance of CIM_Error, the value of CIMStatusCode shall comply with the definition of expected error codes for the CIM-XML operation request. A WBEM client may ignore any <INSTANCE> child elements.

The symbolic names defined in Table 1 do not appear on the wire. They are used here solely for convenient reference to an error in other parts of this document.

Not all methods are expected to return all the status codes listed in Table 1. For intrinsic methods, the relevant clause on each method in this document defines the error codes expected to be returned. For extrinsic methods, 5.4.5 specifies which of the codes in Table 1 can be used.

670 **Table 1 – Status Codes Returned by an <Error> Child element**

Symbolic Name	Code	Definition
CIM_ERR_FAILED	1	A general error occurred that is not covered by a more specific error code.
CIM_ERR_ACCESS_DENIED	2	Access to a CIM resource is not available to the client.
CIM_ERR_INVALID_NAMESPACE	3	The target namespace does not exist.
CIM_ERR_INVALID_PARAMETER	4	One or more parameter values passed to the method are not valid.
CIM_ERR_INVALID_CLASS	5	The specified class does not exist.
CIM_ERR_NOT_FOUND	6	The requested object cannot be found. The operation can be unsupported on behalf of the WBEM server in general or on behalf of an implementation of a management profile.
CIM_ERR_NOT_SUPPORTED	7	The requested operation is not supported on behalf of the WBEM server, or on behalf of a provided class. If the operation is supported for a provided class but is not supported for particular instances of that class, then CIM_ERR_FAILED shall be used.
CIM_ERR_CLASS_HAS_CHILDREN	8	The operation cannot be invoked on this class because it has subclasses.
CIM_ERR_CLASS_HAS_INSTANCES	9	The operation cannot be invoked on this class because one or more instances of this class exist.
CIM_ERR_INVALID_SUPERCLASS	10	The operation cannot be invoked because the specified superclass does not exist.
CIM_ERR_ALREADY_EXISTS	11	The operation cannot be invoked because an object already exists.
CIM_ERR_NO_SUCH_PROPERTY	12	The specified property does not exist.
CIM_ERR_TYPE_MISMATCH	13	The value supplied is not compatible with the type.
CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED	14	The query language is not recognized or supported.
CIM_ERR_INVALID_QUERY	15	The query is not valid for the specified query language.
CIM_ERR_METHOD_NOT_AVAILABLE	16	The extrinsic method cannot be invoked.
CIM_ERR_METHOD_NOT_FOUND	17	The specified extrinsic method does not exist.

Symbolic Name	Code	Definition
CIM_ERR_NAMESPACE_NOT_EMPTY	20	The specified namespace is not empty.
CIM_ERR_INVALID_ENUMERATION_CONTEXT	21	The enumeration identified by the specified context cannot be found, is in a closed state, does not exist, or is otherwise invalid.
CIM_ERR_INVALID_OPERATION_TIMEOUT	22	The specified operation timeout is not supported by the WBEM server.
CIM_ERR_PULL_HAS_BEEN_ABANDONED	23	The Pull operation has been abandoned due to execution of a concurrent CloseEnumeration operation on the same enumeration.
CIM_ERR_PULL_CANNOT_BE_ABANDONED	24	The attempt to abandon a concurrent Pull operation on the same enumeration failed. The concurrent Pull operation proceeds normally.
CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED	25	Using a a filter query in pulled enumerations is not supported by the WBEM server.
CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED	26	The WBEM server does not support continuation on error.
CIM_ERR_SERVER_LIMITS_EXCEEDED	27	The WBEM server has failed the operation based upon exceeding server limits.
CIM_ERR_SERVER_IS_SHUTTING_DOWN	28	The WBEM server is shutting down and cannot process the operation.

5.4.2 Intrinsic Methods

This clause describes the Intrinsic methods defined outside the schema for CIM operations. These methods can only be called on a CIM namespace, rather than on a CIM class or instance.

The notation used in the following subclauses to define the signatures of the intrinsic methods is a pseudo-MOF notation that extends the standard MOF BNF (DSP0004) for describing CIM methods with several pseudo-parameter types enclosed within angle brackets (< and >).

This notation decorates the parameters with pseudo-qualifiers (IN, OUT, OPTIONAL, and NULL) to define their invocation semantics. These qualifiers are for description purposes only within the scope of this document; in particular, a WBEM client shall not specify them in intrinsic method invocations.

This notation uses the IN qualifier to denote that the parameter is an input parameter.

This notation uses the OUT qualifier to denote that the parameter is an output parameter.

A WBEM client may omit an optional parameter by not specifying an <IPARAMVALUE> element for that parameter if the required value is the specified default. It shall not omit a parameter that is not marked as optional. A WBEM server may omit support for an optional parameter. Any attempt to call a method with an optional parameter that is not supported shall return either CIM_ERR_NOT_SUPPORTED or CIM_ERR_INVALID_PARAMETER.

This notation uses the NULL qualifier for parameters whose values can be specified as NULL in a method call. A NULL (unassigned) value for a parameter is specified by an <IPARAMVALUE> or

689 <PARAMVALUE> element with no child element. For parameters without the NULL qualifier, the WBEM
690 client shall specify a value by including a suitable child element for the <IPARAMVALUE> or
691 <PARAMVALUE> element.

692 All parameters shall be uniquely named and shall correspond to a valid parameter name for that method
693 as described by this document. The order of the parameters is not significant.

694 The non-NULL values of intrinsic method parameters or return values modeled as standard CIM types
695 (such as string and Boolean or arrays thereof) are represented as follows:

696 • Simple values use the <VALUE> child element within an <IPARAMETER> element for method
697 parameters or within an <IRETURNVALUE> element for method return values.

698 • Array values use the <VALUE.ARRAY> child element within an <IPARAMETER> element for
699 method parameters or within an <IRETURNVALUE> element for method return values.

700 Table 2 shows how each pseudo-type used by the intrinsic methods shall be mapped to an XML element
701 described in DSP0201 in the context of both a parameter value (child element of <IPARAMVALUE>) and
702 a return value (child element of <IRETURNVALUE>).

703 **Table 2 – Mapping of Intrinsic Method Pseudo-Types to XML Elements**

Type	XML Element
<object>	(VALUE.OBJECT|VALUE.OBJECTWITHLOCALPATH|VALUE.OBJECTWITHPATH)
<class>	CLASS
<instance>	INSTANCE
<className>	CLASSNAME
<namedInstance>	VALUE.NAMEDINSTANCE
<instanceName>	INSTANCENAME
<instancePath>	INSTANCEPATH
<objectWithPath>	VALUE.OBJECTWITHPATH
<instanceWithPath>	VALUE.INSTANCEWITHPATH
<objectName>	(CLASSNAME|INSTANCENAME)
<objectPath>	OBJECTPATH
<propertyValue>	(VALUE|VALUE.ARRAY|VALUE.REFERENCE)
<qualifierDecl>	QUALIFIER.DECLARATION

704 #### 5.4.2.1 GetClass

705 The GetClass operation returns a single CIM class from the target namespace:

```
706     <class> GetClass (
707        [IN] <className> ClassName,
708        [IN,OPTIONAL] boolean LocalOnly = true,
709        [IN,OPTIONAL] boolean IncludeQualifiers = true,
710        [IN,OPTIONAL] boolean IncludeClassOrigin = false,
711        [IN,OPTIONAL,NULL] string PropertyList [] = NULL
712     )
```

713 The `ClassName` input parameter defines the name of the class to be retrieved.

If the `LocalOnly` input parameter is `true`, any CIM elements (properties, methods, and qualifiers), except those added or overridden in the class as specified in the classname input parameter, shall not be included in the returned class. If it is false, no additional filtering is defined.

If the `IncludeQualifiers` input parameter is `true`, all qualifiers for that class (including qualifiers on the class and on any returned properties, methods, or method parameters) shall be included as <QUALIFIER> XML elements in the response. If it is `false`, no <QUALIFIER> XML elements are present in the returned class.

If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on all appropriate elements in the returned class. If it is `false`, no `CLASSORIGIN` attributes are present in the returned class.

If the `PropertyList` input parameter is not `NULL`, the members of the array define one or more property names. The returned class shall not include any properties missing from this list. Note that if `LocalOnly` is specified as `true`, it acts as an additional filter on the set of properties returned. For example, if property `A` is included in `PropertyList` but `LocalOnly` is set to `true` and `A` is not local to the requested class, it is not included in the response. If the `PropertyList` input parameter is an empty array, no properties are included in the response. If the `PropertyList` input parameter is NULL, no additional filtering is defined.

If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise process the request normally. If `PropertyList` contains property names that are invalid for the target class, the WBEM server shall ignore them but otherwise process the request normally.

If GetClass is successful, the return value is a single CIM class that shall include all CIM elements (properties, methods, and qualifiers) defined in or inherited by that class, reduced by any elements excluded as a result of using the `LocalOnly` or `PropertyList` filters.

If GetClass is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses:

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters)
- CIM_ERR_NOT_FOUND (The request CIM class does not exist in the specified namespace.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.2 GetInstance

The GetInstance operation returns a single CIM instance from the target namespace:

```
<instance> GetInstance (
   [IN] <instanceName> InstanceName,
   [IN,OPTIONAL] boolean LocalOnly = true,           (DEPRECATED)
   [IN,OPTIONAL] boolean IncludeQualifiers = false,  (DEPRECATED)
   [IN,OPTIONAL] boolean IncludeClassOrigin = false,
   [IN,OPTIONAL,NULL] string PropertyList [] = NULL
)
```

The `InstanceName` input parameter defines the name of the instance to be retrieved.

DEPRECATION NOTE: With version 1.2 of this document, the `LocalOnly` parameter is DEPRECATED. `LocalOnly` filtering, as defined in 1.1, will not be supported in the next major revision of this document. In version 1.1 of this document, the definition of the `LocalOnly` parameter was incorrectly modified. This change introduced a number of interoperability and backward compatibility problems for WBEM clients using the `LocalOnly` parameter to filter the set of properties returned. The DMTF strongly recommends that WBEM clients set `LocalOnly` to `false` and do not use this parameter to filter the set of properties returned. To minimize the impact of this recommendation on WBEM clients, a WBEM server may choose to treat the value of the `LocalOnly` parameter as `false` for all requests. A WBEM server shall consistently support a single interpretation of the `LocalOnly` parameter. Refer to ANNEX B for additional details.

DEPRECATION NOTE: The use of the `IncludeQualifiers` parameter is DEPRECATED and it may be removed in a future version of this document. The `IncludeQualifiers` parameter definition is ambiguous and when it is set to `true`, WBEM clients cannot be assured that any qualifiers will be returned. A WBEM client should always set `IncludeQualifiers` to `false`. To minimize the impact of this recommendation on WBEM clients, a WBEM server may choose to treat the value of the `IncludeQualifiers` parameter as `false` for all requests. The preferred behavior is to use the class operations to receive qualifier information and not depend on any qualifiers existing in this response. If the `IncludeQualifiers` input parameter is `true`, all qualifiers for that instance (including qualifiers on the instance and on any returned properties) shall be included as <QUALIFIER> XML elements in the response. If it is `false`, no <QUALIFIER> XML elements are present.

If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on all appropriate elements in the returned instance. If it is `false`, no `CLASSORIGIN` attributes are present.

If the `PropertyList` input parameter is not `NULL`, the members of the array define one or more property names. The returned instance shall not include any properties missing from this list. Note that if `LocalOnly` is `true`, this acts as an additional filter on the set of properties returned. For example, if property `A` is included in `PropertyList` but `LocalOnly` is set to `true` and `A` is not local to the requested instance, it is not included in the response. If the `PropertyList` input parameter is an empty array, no properties are included in the response. If the `PropertyList` input parameter is NULL, no additional filtering is defined.

If `PropertyList` contains duplicate property names, the WBEM server shall ignore the duplicates but otherwise process the request normally. If `PropertyList` contains property names that are invalid for the target instance, the WBEM server shall ignore them but otherwise process the request normally.

Properties with the NULL value may be omitted from the response, even if the WBEM client has not requested the exclusion of the property through the `LocalOnly` or `PropertyList` filters. The WBEM client shall interpret such omitted properties as NULL. Note that the WBEM client cannot make any assumptions about properties omitted as a result of using `LocalOnly` or `PropertyList` filters.

If GetInstance is successful, the return value is a single CIM instance with all properties defined in and inherited by its class reduced by any properties excluded as a result of using the `LocalOnly` or `PropertyList` filters and further reduced by any NULL valued properties omitted from the response.

If GetInstance is unsuccessful, the method shall return one of the following status codes where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses:

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)

- CIM_ERR_INVALID_CLASS (The CIM class does not exist in the specified namespace.)
- CIM_ERR_NOT_FOUND (The CIM class does exist, but the requested CIM instance does not exist in the specified namespace.)
- CIM_ERR_FAILED (some other unspecified error occurred)

5.4.2.3 DeleteClass

The DeleteClass operation deletes a single CIM class from the target namespace:

```
void DeleteClass (
   [IN] <className> ClassName
)
```

The `ClassName` input parameter defines the name of the class to be deleted.

If DeleteClass is successful, the WBEM server removes the specified class, including any subclasses and any instances. The operation shall fail if any one of these objects cannot be deleted.

If DeleteClass is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses:

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_FOUND (The CIM class to be deleted does not exist.)
- CIM_ERR_CLASS_HAS_CHILDREN (The CIM class has one or more subclasses that cannot be deleted.)
- CIM_ERR_CLASS_HAS_INSTANCES (The CIM class has one or more instances that cannot be deleted.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.4 DeleteInstance

The DeleteInstance operation deletes a single CIM instance from the target namespace.

```
void DeleteInstance (
   [IN] <instanceName> InstanceName
)
```

The `InstanceName` input parameter defines the name (model path) of the instance to be deleted.

Deleting the instance may or may not cause the automatic deletion of additional instances. For example, the deletion of an instance may cause the automatic deletion of all associations that reference that instance. Or the deletion of an instance may cause the automatic deletion of instances (and their associations) that have a Min(1) relationship to that instance.

If DeleteInstance is successful, the WBEM server removes the specified instance.

If DeleteInstance is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_CLASS (The CIM class does not exist in the specified namespace.)
- CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified instance, if provided.)
- CIM_ERR_NOT_FOUND (The CIM class does exist, but the requested CIM instance does not exist in the specified namespace.)
- CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other unspecified error occurred.)

5.4.2.5 CreateClass

The CreateClass operation creates a single CIM class in the target namespace. The class shall not already exist:

```
void CreateClass (
   [IN] <class> NewClass
)
```

The `NewClass` input parameter defines the new class. The proposed definition shall be a correct class definition according to DSP0004.

In processing the creation of the new class, the WBEM server shall conform to the following rules:

- The server shall ignore any CLASSORIGIN and PROPAGATED XML attributes in the new class.
- If the new class has no superclass, the `NewClass` parameter defines a new superclass. The server shall ensure that all properties and methods of the new class have a CLASSORIGIN attribute whose value is the name of the new class.
- If the new class has a superclass, the `NewClass` parameter defines a new subclass of that superclass. The superclass shall exist. The server shall ensure that the following conditions are met:
  - Any properties, methods, or qualifiers in the subclass not defined in the superclass are created as new elements of the subclass. In particular, the server shall set the CLASSORIGIN XML attribute on the new properties and methods to the name of the subclass and ensure that all others preserve their CLASSORIGIN attribute value from that defined in the superclass.
  - If a property is defined in the superclass and in the subclass, the value assigned to that property in the subclass (including NULL) becomes the default value of the property for the subclass.
  - If a property or method of the superclass is not specified in the subclass, then it is inherited without modification by the subclass.

881 – Any qualifiers defined in the superclass with a TOSUBCLASS attribute value of `true` shall
882 appear in the resulting subclass. Qualifiers in the superclass with a TOSUBCLASS
883 attribute value of `false` shall not be propagated to the subclass.

884 – Any qualifier propagated from the superclass cannot be modified in the subclass if the
885 OVERRIDABLE attribute of that qualifier is set to `false` in the superclass. It is a client
886 error to specify such a qualifier in the new class with a definition different than that in the
887 superclass (where definition encompasses the name, type, and flavor attribute settings of
888 the <QUALIFIER> XML element and the value of the qualifier).

889 If CreateClass is successful, the WBEM server creates the specified class.

890 If CreateClass is unsuccessful, this method shall return one of the following status codes, where the error
891 returned is the first applicable error in the list, starting with the first element and working down. Any
892 additional method-specific interpretation of the error is enclosed in parentheses.

893 • CIM_ERR_ACCESS_DENIED

894 • CIM_ERR_NOT_SUPPORTED

895 • CIM_ERR_INVALID_NAMESPACE

896 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
897 incorrect parameters)

898 • CIM_ERR_ALREADY_EXISTS (The CIM class already exists.)

899 • CIM_ERR_INVALID_SUPERCLASS (The putative CIM class declares a non-existent
900 superclass.)

901 • CIM_ERR_FAILED (Some other unspecified error occurred.)

902 #### 5.4.2.6 CreateInstance

903 The CreateInstance operation creates a single CIM Instance in the target namespace. The instance shall
904 not already exist:

```
905     <instanceName> CreateInstance (
906        [IN] <instance> NewInstance
907     )
```

908 **DEPRECATION NOTE:** The use of qualifiers on instances is DEPRECATED and may be removed in a
909 future version of this document. A WBEM client cannot rely on any qualifiers included in the
910 `NewInstance` to have any impact on the operation. It is recommended that the WBEM server ignore any
911 qualifiers included in the instance. The `NewInstance` input parameter defines the new instance. The
912 proposed definition shall be a correct instance definition for the underlying CIM class according to
913 DSP0004.

914 In creating the new instance, the WBEM server shall conform to the following rules and ensure that they
915 are applied:

916 • The server shall ignore any CLASSORIGIN and PROPAGATED XML attributes in the
917 `NewInstance`.

918 • **DEPRECATED.** Any qualifiers in the instance not defined in the class are created as new
919 elements of the instance.

920 • All properties of the instance preserve their CLASSORIGIN attribute value from that defined in
921 the class.

922 • The designated initial value for any property in the CIM instance to be created shall be the
923 property value (including NULL) specified in the `NewInstance` parameter, or if the property is

not specified in the `NewInstance` parameter, the default value (including NULL) defined in the property declaration, or if the property does not define a default value, there is no designated initial value for the property.

If there is a designated initial value for a property, the server shall either initialize the property to that value, or reject the request. If there is no designated initial value for a property, the server may initialize the property to any value (including NULL). Further considerations for accepting or rejecting creation requests based on the properties requested to be initialized are out of scope for this document; CIM model definitions are expected to cover that.

- If the `NewInstance` parameter specifies properties that are not exposed by the class specified in the `NewInstance` parameter, the server shall reject the request.
- **DEPRECATION NOTE:** Use of the TOINSTANCE attribute is DEPRECATED. Servers may choose to ignore TOINSTANCE. Servers that do not ignore TOINSTANCE shall interpret it so that any qualifiers defined in the class with a TOINSTANCE attribute value of `true` appear in the instance. Qualifiers in the class with a value of `false` shall not be propagated to the instance.
- **DEPRECATED.** Any Qualifier propagated from the class cannot be modified in the instance if the OVERRIDABLE attribute of that qualifier is set to `false` in the class. It is a client error to specify such a qualifier in the `NewInstance` with a definition different than that in the class (where definition encompasses the name, type, and flavor attribute settings of the <QUALIFIER> XML element and the value of the qualifier).

If CreateInstance is successful, the new CIM instance has been created as described in this subclause, and the return value defines the object path of the new CIM instance relative to the target namespace created by the WBEM server (that is, the model path as defined by DSP0004). It is returned if one or more of the new keys of the instance are dynamically allocated during creation rather than specified in the request.

If CreateInstance is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_CLASS (The CIM class for the new instance does not exist.)
- CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified instance, if provided.)
- CIM_ERR_ALREADY_EXISTS (The CIM instance already exists.)
- CIM_ERR_FAILED (This operation is not supported for the specified instance or some other unspecified error occurred.)

5.4.2.7 ModifyClass

The ModifyClass operation modifies an existing CIM class in the target namespace. The class shall already exist:

```
void ModifyClass (
   [IN] <class> ModifiedClass
```

968)

969 The `ModifiedClass` input parameter defines the set of changes to be made to the current class
970 definition, which shall be correct amendments to the CIM class as defined by DSP0004.

971 In modifying the class, the WBEM server shall conform to the following rules:

972 • The WBEM server shall ignore any CLASSORIGIN and PROPAGATED XML attributes in the
973 `ModifiedClass`.

974 • If the modified class has no superclass, the ModifiedClass parameter defines modifications to a
975 superclass. The server shall ensure that the following conditions are met:

976 – All properties and methods of the modified class have a CLASSORIGIN attribute whose
977 value is the name of this class.

978 – Any properties, methods, or qualifiers in the existing class definition that do not appear in
979 the `ModifiedClass` parameter are removed from the resulting modified class.

980 • If the modified class has a superclass, the `ModifiedClass` parameter defines modifications to
981 a subclass of that superclass. The superclass shall exist, and the client shall not change the
982 name of the superclass in the modified subclass. The server shall ensure that the following
983 conditions are met:

984 – Any properties, methods, or qualifiers in the subclass not defined in the superclass are
985 created as elements of the subclass. In particular, the server shall set the CLASSORIGIN
986 attribute on the new properties and methods to the name of the subclass and shall ensure
987 that all other others preserve their CLASSORIGIN attribute value from that defined in the
988 superclass.

989 – Any property, method, or qualifier previously defined in the subclass but not defined in the
990 superclass, and which is not present in the `ModifiedClass` parameter, is removed from
991 the subclass.

992 – If a property is specified in the `ModifiedClass` parameter, the value assigned to that
993 property (including NULL) becomes the default value of the property for the subclass.

994 – If a property or method of the superclass is not specified in the subclass, then the subclass
995 inherits it without modification. Any previous changes to such an element in the subclass
996 are lost.

997 – If a qualifier in the superclass is not specified in the subclass and the qualifier is defined in
998 the superclass with a TOSUBCLASS attribute value of `true`, then the qualifier shall still be
999 present in the resulting modified subclass. A propagated qualifier cannot be removed from
1000 a subclass.

1001 – Any qualifier propagated from the superclass cannot be modified in the subclass if the
1002 OVERRIDABLE attribute of that qualifier is set to `false` in the superclass. It is a client
1003 error to specify such a qualifier in the ModifiedClass with a definition different than that in
1004 the superclass (where definition encompasses the name, type, and flavor attribute settings
1005 of the <QUALIFIER> XML element and the value of the qualifier).

1006 – Any qualifiers defined in the superclass with a TOSUBCLASS attribute value of `false`
1007 shall not be propagated to the subclass.

1008 If ModifyClass is successful, the WBEM server updates the specified class. The request to modify the
1009 class shall fail if the server cannot consistently update any existing subclasses or instances of that class.

1010 If ModifyClass is unsuccessful, this method shall return one of the following status codes, where the error
1011 returned is the first applicable error in the list, starting with the first element and working down. Any
1012 additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_FOUND (The CIM class does not exist.)
- CIM_ERR_INVALID_SUPERCLASS (The putative CIM class declares a non-existent or incorrect superclass.)
- CIM_ERR_CLASS_HAS_CHILDREN (The modification could not be performed because the subclasses of the class could not be updated consistently.)
- CIM_ERR_CLASS_HAS_INSTANCES (The modification could not be performed because the instances of the class could not be updated consistently.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.8 ModifyInstance

The ModifyInstance operation modifies an existing CIM instance in the target namespace. The instance shall already exist:

```
void ModifyInstance (
   [IN] <namedInstance> ModifiedInstance,
   [IN, OPTIONAL] boolean IncludeQualifiers = true,     (DEPRECATED)
   [IN, OPTIONAL, NULL] string PropertyList[] = NULL
)
```

The `ModifiedInstance` input parameter identifies the name of the instance to be modified and provides the new property values.

DEPRECATION NOTE: Use of the `IncludeQualifiers` parameter is DEPRECATED, and it may be removed in a future version of this document. The behavior of the `IncludeQualifiers` parameter is not specified. A WBEM client cannot rely on `IncludeQualifiers` to have any impact on the operation. It is recommended that the WBEM server ignore any qualifiers included in `ModifiedInstance`. If the `IncludeQualifiers` input parameter is `true`, the qualifiers are modified as specified in `ModifiedInstance`. If the parameter is `false`, qualifiers in `ModifiedInstance` are ignored and no qualifiers are explicitly modified.

The set of properties designated to be modified shall be determined as follows:

If the `PropertyList` input parameter is not NULL, the members of the array define one or more property names. The properties specified in `PropertyList` are designated to be modified. Properties of the `ModifiedInstance` that are missing from `PropertyList` are not designated to be modified. If `PropertyList` is an empty array, no properties are designated to be modified. If `PropertyList` is NULL, the properties of `ModifiedInstance` with values different from the current values in the instance are designated to be modified.

If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise process the request normally. If `PropertyList` contains property names that are invalid for the instance to be modified, the WBEM server shall reject the request.

If a property is designated to be modified, the WBEM server shall either modify the property, or reject the request. The server shall reject modification requests for key properties. Further considerations for accepting or rejecting modification requests based on the properties requested to be modified are out of

scope for this document; CIM model definitions are expected to cover that. Note that the WRITE qualifier on a property is considered to be in the area of CIM models; specifically, a value of True for the WRITE qualifier does not guarantee modifiability of that property, and a value of False does not prevent modifiability.

If a property is not designated to be modified, the server shall not modify its value. However, note that properties may change their values as a result of other changes.

In modifying the instance, the WBEM server shall conform to the following rules and ensure their application:

- The server shall ignore any CLASSORIGIN and PROPAGATED attributes in the ModifiedInstance.
- The class shall exist, and the client shall not change its name in the instance to be modified.
- **DEPRECATED.** Any qualifiers in the instance not defined in the class are created as new elements of the instance if `IncludeQualifiers` is `true`.
- All properties of the instance to be modified preserve their CLASSORIGIN attribute value from that defined in the class.
- **DEPRECATED.** Any qualifier previously defined in the instance to be modified but not defined in the class, and which is not present in the `ModifiedInstance` parameter, is removed from the instance if `IncludeQualifiers` is `true`.
- If a property is to be modified as previously defined, the designated new value for that property in the CIM instance shall be the property value (including NULL) specified in the `ModifiedInstance` parameter, or if the property is not specified in the `ModifiedInstance` parameter, the default value (including NULL) defined in the property declaration, or if the property does not define a default value, there is no designated new value for the property.

  If there is a designated new value for a property, the server shall either update the property to that value, or reject the request. If there is no designated new value for a property, the server may update the property to any value (including NULL). Further determinations about this decision are out of scope for this document; CIM model definitions are expected to cover that..
- **DEPRECATION NOTE:** The use of the TOINSTANCE qualifier attribute is DEPRECATED. Servers may choose to ignore TOINSTANCE. Servers that do not ignore TOINSTANCE shall interpret it so that any qualifiers defined in the class with a TOINSTANCE attribute value of true appear in the instance. A propagated qualifier cannot be removed from an instance. qualifiers in the class with a TOINSTANCE attribute value of `false` shall not be propagated to the instance
- **DEPRECATED.** Any qualifier propagated from the class cannot be modified in the instance if the OVERRIDABLE attribute of that qualifier is set to false in the class. It is a client error to specify such a qualifier in `ModifiedInstance` with a definition different than that in the class (where definition encompasses the name, type, and flavor attribute settings of the <QUALIFIER> XML element and the value of the qualifier).

If ModifyInstance is successful, the specified CIM instance has been updated as described in this subclause.

If ModifyInstance is unsuccessful, the specified Instance is not updated, and the method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)
- CIM_ERR_INVALID_NAMESPACE

- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters and invalid properties to be modified)
- CIM_ERR_INVALID_CLASS (The CIM class of the instance to be modified does not exist.)
- CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified instance, if provided.)
- CIM_ERR_NOT_FOUND (The CIM instance to be modified does not exist.)
- CIM_ERR_FAILED (This operation is not supported for the specified instance or some other unspecified error occurred, including a request for non-writable properties to be modified or a property that cannot be modified at this time.)

5.4.2.9 EnumerateClasses

The EnumerateClasses operation enumerates subclasses of a CIM class in the target namespace:

```
<class>* EnumerateClasses (
   [IN,OPTIONAL,NULL] <className> ClassName=NULL,
   [IN,OPTIONAL] boolean DeepInheritance = false,
   [IN,OPTIONAL] boolean LocalOnly = true,
   [IN,OPTIONAL] boolean IncludeQualifiers = true,
   [IN,OPTIONAL] boolean IncludeClassOrigin = false
)
```

The `ClassName` input parameter defines the class that is the basis for the enumeration.

If the `DeepInheritance` input parameter is `true`, all subclasses of the specified class should be returned. If the `ClassName` input parameter is absent, this implies that all classes in the target namespace should be returned. If `DeepInheritance` is `false`, only immediate child subclasses are returned. If the `ClassName` input parameter is NULL, this implies that all top-level classes (that is, classes with no superclass) in the target namespace should be returned. This definition of `DeepInheritance` applies only to the EnumerateClasses and EnumerateClassName operations.

If the `LocalOnly` input parameter is `true`, any CIM elements (properties, methods, and qualifiers) except those added or overridden in the class as specified in the classname input parameter shall not be included in the returned class. If it is `false`, this parameter defines no additional filtering.

If the `IncludeQualifiers` input parameter is `true`, all qualifiers for each class (including qualifiers on the class and on any returned properties, methods, or method parameters) shall be included as <QUALIFIER> XML elements in the response. If it is `false`, no <QUALIFIER> XML elements are present.

If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on all appropriate elements in each returned class. If it is `false`, no `CLASSORIGIN` attributes are present.

If EnumerateClasses is successful, the method returns zero or more classes that meet the required criteria. These classes shall include all CIM elements (properties, methods, and qualifiers) defined in or inherited by each class, reduced by any elements excluded as a result of using the `LocalOnly` filter.

If EnumerateClasses is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED

- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_CLASS (The CIM class for this enumeration does not exist.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.10 EnumerateClassNames

The EnumerateClassNames operation enumerates the names of subclasses of a CIM class in the target namespace:

```
<className>* EnumerateClassNames (
   [IN,OPTIONAL,NULL] <className> ClassName = NULL,
   [IN,OPTIONAL] boolean DeepInheritance = false
)
```

The `ClassName` input parameter defines the class that is the basis for the enumeration.

If the `DeepInheritance` input parameter is `true`, the names of all subclasses of the specified class should be returned. If the ClassName input parameter is absent, this implies that the names of all classes in the target namespace should be returned. If DeepInheritance is `false`, only the names of immediate child subclasses are returned. If the `ClassName` input parameter is NULL, this implies that the names of all top-level classes (that is, classes with no superclass) in the target namespace should be returned. This definition of `DeepInheritance` applies only to the EnumerateClasses and EnumerateClassName operations.

If EnumerateClassNames is successful, the method returns zero or more names of classes that meet the requested criteria.

If EnumerateClassNames is unsuccessful, this method returns one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not exist.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.11 EnumerateInstances (DEPRECATED)

The EnumerateInstances operation enumerates instances of a CIM class in the target namespace, including instances in the class and any subclasses in accordance with the polymorphic nature of CIM objects:

```
<namedInstance>* EnumerateInstances (
   [IN] <className> ClassName,
   [IN,OPTIONAL] boolean LocalOnly = true,           (DEPRECATED)
   [IN,OPTIONAL] boolean DeepInheritance = true,
   [IN,OPTIONAL] boolean IncludeQualifiers = false,  (DEPRECATED)
```

```
        [IN,OPTIONAL] boolean IncludeClassOrigin = false,
        [IN,OPTIONAL,NULL] string PropertyList [] = NULL
    )
```

DEPRECATION NOTE: The EnumerateInstances operation has been deprecated in version 1.4 of this document. Use OpenEnumerateInstances instead (see 5.4.2.24.3).

The `ClassName` input parameter defines the class that is the basis for the enumeration.

DEPRECATION NOTE: With version 1.2of this document, the `LocalOnly` parameter is DEPRECATED. `LocalOnly` filtering, as defined in 1.1, will not be supported in the next major revision of this document. In version 1.1of this document, the definition of the `LocalOnly` parameter was incorrectly modified. This change introduced a number of interoperability and backward compatibility problems for WBEM clients using the `LocalOnly` parameter to filter the set of properties returned. The DMTF strongly recommends that WBEM clients set `LocalOnly` to `false` and do not use this parameter to filter the set of properties returned. To minimize the impact of this recommendation on WBEM clients, a WBEM server may choose to treat the value of the `LocalOnly` parameter as `false` for all requests. A WBEM server shall consistently support a single interpretation of the `LocalOnly` parameter. Refer to ANNEX B for details.

If the `DeepInheritance` input parameter is `false`, each returned instance shall not include any properties added by subclasses of the specified class. If it is `true`, no additional filtering is defined.

DEPRECATION NOTE: The use of the `IncludeQualifiers` parameter is DEPRECATED and it may be removed in a future version of this document. The definition of `IncludeQualifiers` is ambiguous and when this parameter is set to `true`, WBEM clients cannot be assured that any qualifiers will be returned. A WBEM client should always set this parameter to `false`. To minimize the impact of this recommendation on WBEM clients, a WBEM server may choose to treat the value of `IncludeQualifiers` as `false` for all requests. The preferred behavior is to use the class operations to receive qualifier information and not depend on any qualifiers in this response. If the `IncludeQualifiers` input parameter is `true`, all qualifiers for the instance, (including qualifiers on the instance and on any returned properties, shall be included as <QUALIFIER> XML elements in the response. If it is `false`, no <QUALIFIER> XML elements are present in the returned instance.

If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on all appropriate elements in each returned Instance. If it is `false`, no `CLASSORIGIN` attributes are present.

If the `PropertyList` input parameter is not NULL, the members of the array define one or more property names of the designated class. This definition may include inherited property names or property names explicitly defined in the designated class. However, it may not include property names added in subclasses of the designated class. Each returned instance shall not include any properties missing from this list. Note that `PropertyList` acts as an additional filter on the properties defined by the `LocalOnly` and `DeepInheritance` input parameters; if `PropertyList` includes a property name that is not in the set defined by the `LocalOnly` and `DeepInheritance` combination, the element for the property shall not be included in the returned instances. If `PropertyList` is an empty array, no properties are included in the returned instances. If `PropertyList` is NULL, no additional filtering is defined.

If `PropertyList` contains duplicate property names, the WBEM server shall ignore the duplicates but otherwise process the request normally. If `PropertyList` contains property names that are invalid for a target instance, the WBEM server shall ignore them for that instance but otherwise process the request normally.

Properties with the NULL value may be omitted from the response, even if the WBEM client has not requested the exclusion of the property through the `LocalOnly`, `DeepInheritance`, or `PropertyList` filters. The WBEM client shall interpret such omitted properties as NULL. Note that the WBEM client

cannot make any assumptions about properties omitted as a result of using any `LocalOnly`, `DeepInheritance`, or `PropertyList` filters.

If EnumerateInstances is successful, the method returns zero or more `<namedInstance>` items representing named instances that meet the required criteria. These instances shall have all properties defined in and inherited by their respective classes, reduced by any properties excluded as a result of using the `LocalOnly`, `DeepInheritance`, or `PropertyList` filters and further reduced by any NULL-valued properties omitted from the response.

If EnumerateInstances is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not exist.)
- CIM_ERR_NOT_SUPPORTED (This operation is not supported for the specified class and all of its subclasses, if provided.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.12 EnumerateInstanceNames (DEPRECATED)

The EnumerateInstanceNames operation enumerates the names (model paths) of the instances of a CIM class in the target namespace, including instances in the class and any subclasses in accordance with the polymorphic nature of CIM objects:

```
<instanceName>* EnumerateInstanceNames (
   [IN] <className> ClassName
)
```

DEPRECATION NOTE: The EnumerateInstanceNames operation has been deprecated in version 1.4 of this document. Use OpenEnumerateInstancePaths instead (see 5.4.2.24.4).

The `ClassName` input parameter defines the class that is the basis for the enumeration.

If EnumerateInstanceNames is successful, the method returns zero or more `<instanceName>` items representing instance names (referred to in DSP0004 as a model path) that meet the requested criteria. The `<instanceName>` items shall specify the class from which the instance is instantiated, not any of its superclasses. Note that this class may be different from the class specified as input.

If EnumerateInstanceNames is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)

- CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not exist.)
- CIM_ERR_NOT_SUPPORTED (This operation is not supported for the specified class and all of its subclasses, if provided.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.13 ExecQuery (DEPRECATED)

The ExecQuery operation executes a query against the target namespace:

```
<object>* ExecQuery (
   [IN] string QueryLanguage,
   [IN] string Query
)
```

DEPRECATION NOTE: The ExecQuery operation has been deprecated in version 1.4 of this document. Use OpenQueryInstances instead (see 5.4.2.24.14).

The `QueryLanguage` input parameter defines the query language in which the query parameter is expressed.

The `Query` input parameter defines the query to be executed. The results of the query shall be constrained to contain only CIM classes that exist in the target namespace or CIM instances whose classes exist in the target namespace. Note that any instances in the result set may or may not exist in any namespace. Note that for query languages supporting select-lists and from-clauses, this implies that all select-list entries resolve to disjoint properties exposed by one CIM class named in the from-clause. This rule does not prevent such queries from using joins.

Neither the query language nor the format of the query is defined by this document. It is anticipated that query languages will be submitted to the DMTF as separate proposals.

WBEM servers can declare which query languages they support (if any) using a mechanism defined in 7.5.

If ExecQuery is successful, the method returns zero or more `<object>` items representing CIM classes or instances that correspond to the results of the query.

If ExecQuery is unsuccessful, the method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested query language is not recognized.)
- CIM_ERR_INVALID_QUERY (The query is not a valid query in the specified query language.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.14 Associators (PARTLY DEPRECATED)

The Associators operation enumerates CIM objects (classes or instances) associated with a particular source CIM object:

```
<objectWithPath>* Associators (
   [IN] <objectName> ObjectName,
   [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
   [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
   [IN,OPTIONAL,NULL] string Role = NULL,
   [IN,OPTIONAL,NULL] string ResultRole = NULL,
   [IN,OPTIONAL] boolean IncludeQualifiers = false,        (DEPRECATED)
   [IN,OPTIONAL] boolean IncludeClassOrigin = false,
   [IN,OPTIONAL,NULL] string PropertyList [] = NULL
)
```

DEPRECATION NOTE: The Associators operation for instances has been deprecated in version 1.4 of this document. Use OpenAssociatorInstances instead (see 5.4.2.24.7). The Associators operation for classes remains undeprecated.

The `ObjectName` input parameter defines the source CIM object whose associated objects are to be returned. This may be either a class name or instance name (model path).

The `AssocClass` input parameter, if not `NULL`, shall be a valid CIM association class name. It acts as a filter on the returned set of objects by mandating that each returned object shall be associated to the source object through an instance of this class or one of its subclasses.

The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the returned set of objects by mandating that each returned object shall be either an instance of this class (or one of its subclasses) or be this class (or one of its subclasses).

The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned set of objects by mandating that each returned object shall be associated with the source object through an association in which the source object plays the specified role. That is, the name of the property in the association class that refers to the source object shall match the value of this parameter.

The `ResultRole` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned set of objects by mandating that each returned object shall be associated to the source object through an association in which the returned object plays the specified role. That is, the name of the property in the association class that refers to the returned object shall match the value of this parameter.

DEPRECATION NOTE: The use of the `IncludeQualifiers` parameter is DEPRECATED and it may be removed in a future version of this document. The preferred behavior is to use the class operations to receive qualifier information and not depend on any qualifiers in this response. If `IncludeQualifiers` is `true`, all qualifiers for each object (including qualifiers on the object and on any returned properties) shall be included as <QUALIFIER> XML elements in the response. If it is `false`, no <QUALIFIER> XML elements are present.

If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on all appropriate elements in each returned object. If it is `false`, no `CLASSORIGIN` attributes are present.

If the `PropertyList` input parameter is not NULL, the members of the array define one or more property names. Each returned object shall not include any properties missing from this list. If `PropertyList` is an empty array, no properties are included in each returned object. If it is NULL, no additional filtering is defined.

If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise process the request normally. If `PropertyList` contains property names that are invalid for a target object, the WBEM server shall ignore them for that object but otherwise process the request normally.Clients should not explicitly specify properties in the `PropertyList` parameter unless they specify a non-NULL value for the `ResultClass` parameter.

If instances are returned, properties with the NULL value may be omitted from the response, even if the WBEM client has not requested the exclusion of the through the `PropertyList` filter. The WBEM client shall interpret such omitted properties as NULL. Note that the WBEM client cannot make any assumptions about properties omitted as a result of using the `PropertyList` filter. If classes are returned, the WBEM server cannot make this choice, and only the WBEM client can cause properties to be excluded by using the `PropertyList` filter.

If Associators is successful, the method returns zero or more `<objectWithPath>` items representing CIM classes or instances meeting the requested criteria. Because it is possible for CIM objects from different hosts or namespaces to be associated, each returned object includes location information. If the `ObjectName` refers to a class, then classes are returned. These classes shall have all CIM elements (properties, methods, and qualifiers) defined in and inherited by that class, reduced by any properties excluded as a result of using the `PropertyList` filter. If the `ObjectName` refers to an instance, then instances are returned. These instances shall have all properties defined in and inherited by its class, reduced by any properties excluded as a result of using the `PropertyList` filter and further reduced by any NULL valued properties omitted from the response.

If Associators is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified instance, if provided.)
- CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other unspecified error occurred.)

5.4.2.15 AssociatorNames (PARTLY DEPRECATED)

The AssociatorNames operation enumerates the names of CIM Objects (classes or instances) that are associated with a particular source CIM object:

```
<objectPath>* AssociatorNames (
   [IN] <objectName> ObjectName,
   [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
   [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
   [IN,OPTIONAL,NULL] string Role = NULL,
   [IN,OPTIONAL,NULL] string ResultRole = NULL
)
```

DEPRECATION NOTE: The AssociatorNames operation has been deprecated in version 1.4 of this document. Use OpenAssociatorInstancePaths instead (see 5.4.2.24.8). The AssociatorNames operation for classes remains undeprecated.

The `ObjectName` input parameter defines the source CIM object whose associated names are to be returned. This is either a class or instance name (model path).

The `AssocClass` input parameter, if not `NULL`, shall be a valid CIM association class name. It acts as a filter on the returned set of names by mandating that each returned name identify an object that shall be associated to the source object through an instance of this class or one of its subclasses.

The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the returned set of names by mandating that each returned name identify an object that shall be either an instance of this class (or one of its subclasses) or be this class (or one of its subclasses).

The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned set of names by mandating that each returned name identify an object that shall be associated to the source object through an association in which the source object plays the specified role. That is, the name of the property in the association class that refers to the source object shall match the value of this parameter.

The `ResultRole` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned set of names by mandating that each returned name identify an object that shall be associated to the source object through an association in which the named returned object plays the specified role. That is, the name of the property in the association class that refers to the returned object shall match the value of this parameter.

If AssociatorNames is successful, the method returns zero or more `<objectPath>` items representing CIM class paths or instance paths meeting the requested criteria. Because CIM objects from different hosts or namespaces can be associated, each returned object includes location information. If the `ObjectName` refers to a class path, then class paths are returned. Otherwise, the `ObjectName` refers to an instance path, and instance paths are returned.

If AssociatorNames is unsuccessful, one of the following status codes shall be returned by this method, where the first applicable error in the list (starting with the first element of the list, and working down) is the error returned. Any additional method-specific interpretation of the error is given in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters)
- CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified instance, if provided.)
- CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other unspecified error occurred.)

5.4.2.16 References (PARTLY DEPRECATED)

The References operation enumerates the association objects that refer to a particular target CIM object (class or instance).

```
<objectWithPath>* References (
   [IN] <objectName> ObjectName,
   [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
   [IN,OPTIONAL,NULL] string Role = NULL,
   [IN,OPTIONAL] boolean IncludeQualifiers = false,     (DEPRECATED)
   [IN,OPTIONAL] boolean IncludeClassOrigin = false,
```

```
        [IN,OPTIONAL,NULL] string PropertyList [] = NULL
    )
```

DEPRECATION NOTE: The References operation has been deprecated in version 1.4 of this document. Use OpenReferenceInstances instead (see 5.4.2.24.5). The References operation for classes remains undeprecated.

The `ObjectName` input parameter defines the target CIM object whose referring objects are to be returned. This is either a class or instance name (model path).

The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the returned set of objects by mandating that each returned object shall be an instance of this class (or one of its subclasses) or this class (or one of its subclasses).

The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned set of objects by mandating that each returned object shall refer to the target object through a property with a name that matches the value of this parameter.

DEPRECATION NOTE: The use of the `IncludeQualifiers` parameter is DEPRECATED and it may be removed in a future version of this document. The preferred behavior is to use the class operations to receive qualifier information and not depend on any qualifiers in this response. If `IncludeQualifiers` is `true`, all qualifiers for each object (including qualifiers on the object and on any returned properties) shall be included as <QUALIFIER> XML elements in the response. If this parameter is `false`, no <QUALIFIER> XML elements are present in each returned Object.

If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on all appropriate elements in each returned object. If it is `false`, no `CLASSORIGIN` attributes are present.

If the `PropertyList` input parameter is not NULL, the members of the array define one or more property names. Each returned object shall not include any properties missing from this list. If `PropertyList` is an empty array, no properties are included in each returned object. If `PropertyList` is NULL, no additional filtering is defined.

If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise process the request normally. If `PropertyList` contains property names that are invalid for a target object, the WBEM server shall ignore them for that object but otherwise process the request normally.

Clients should not explicitly specify properties in the `PropertyList` parameter unless they specify a non-NULL value for the `ResultClass` parameter.

If instances are returned, properties with the NULL value may be omitted from the response, even if the WBEM client has not requested the exclusion of the property through the `PropertyList` filter. The WBEM client must interpret such omitted properties as NULL. Note that the WBEM client cannot make any assumptions about properties omitted as a result of using the `PropertyList` filter. If classes are returned, the WBEM server cannot make this choice, and only the WBEM client can cause properties to be excluded by using the `PropertyList` filter.

If References is successful, the method returns zero or more `<objectWithPath>` items representing CIM classes or instances meeting the requested criteria. Because CIM objects from different hosts or namespaces can be associated, each returned object includes location information. If the `ObjectName` refers to a class, then classes are returned. These classes shall have all CIM elements (properties, methods, and qualifiers) defined in and inherited by that class, reduced by any properties excluded as a result of using the `PropertyList` filter. If the `ObjectName` refers to an instance, then instances are returned. These instances shall have all properties defined in and inherited by their respective classes, reduced by any properties excluded as a result of using the `PropertyList` filter and further reduced by any NULL valued properties omitted from the response.

If References is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified instance, if provided.)
- CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other unspecified error occurred.)

5.4.2.17 ReferenceNames (PARTLY DEPRECATED)

The ReferenceNames operation enumerates the association objects that refer to a particular target CIM object (class or instance):

```
<objectPath>* ReferenceNames (
   [IN] <objectName> ObjectName,
   [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
   [IN,OPTIONAL,NULL] string Role = NULL
)
```

DEPRECATION NOTE: The ReferenceNames operation has been deprecated in version 1.4 of this document. Use OpenReferenceInstancePaths instead (see 5.4.2.24.6). The ReferenceNames operation for classes remains undeprecated.

The `ObjectName` input parameter defines the target CIM object with the referring object names to be returned. It may be either a class or an instance name (model path).

The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the returned set of object names by mandating that each returned Object Name identify an instance of this class (or one of its subclasses) or this class (or one of its subclasses).

The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned set of object names by mandating that each returned object name shall identify an object that refers to the target instance through a property with a name that matches the value of this parameter.

If ReferenceNames is successful, the method returns zero or more `<objectPath>` items representing CIM class paths or instance paths meeting the requested criteria. Because CIM objects from different hosts or namespaces can be associated, each returned object includes location information. If the `ObjectName` refers to a class path, then class paths are returned. Otherwise, the `ObjectName` refers to an instance path, and instance paths are returned.

If ReferenceNames is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)
- CIM_ERR_INVALID_NAMESPACE

- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified instance, if provided.)
- CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other unspecified error occurred.)

5.4.2.18 GetProperty (DEPRECATED)

The GetProperty operation retrieves a single property value from a CIM instance in the target namespace:

```
<propertyValue> GetProperty (
   [IN] <instanceName> InstanceName,
   [IN] string PropertyName
)
```

DEPRECATION NOTE: The GetProperty operation has been deprecated in version 1.4 of this document. Use GetInstance instead (see 5.4.2.2).

The `InstanceName` input parameter specifies the name of the instance (model path) from which the property value is requested.

The `PropertyName` input parameter specifies the name of the property with the value to be returned.

If GetProperty is successful, the return value specifies the value of the requested property. If the value is NULL, no element is returned.

If GetProperty is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_CLASS (The CIM class does not exist in the specified namespace.)
- CIM_ERR_NOT_FOUND (The CIM class exists, but the requested CIM instance does not exist in the specified namespace.)
- CIM_ERR_NO_SUCH_PROPERTY (The CIM instance exists, but the requested property does not.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.19 SetProperty (DEPRECATED)

The SetProperty operation sets a single property value in a CIM instance in the target namespace:

```
void SetProperty (
   [IN] <instanceName> InstanceName,
   [IN] string PropertyName,
   [IN,OPTIONAL,NULL] <propertyValue> NewValue = NULL
)
```

DEPRECATION NOTE: The SetProperty operation has been deprecated in version 1.4 of this document. Use ModifyInstance instead (see 5.4.2.8).

The `InstanceName` input parameter specifies the name of the instance (model path) with the property value to be updated.

The `PropertyName` input parameter specifies the name of the property with the value to be updated.

The `NewValue` input parameter specifies the new value for the property (which may be NULL).

If SetProperty is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_CLASS (The CIM class does not exist in the specified namespace.)
- CIM_ERR_NOT_FOUND (The CIM class exists, but the requested CIM instance does not exist in the specified namespace.)
- CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified instance, if provided.)
- CIM_ERR_NO_SUCH_PROPERTY (The CIM instance exists, but the requested property does not.)
- CIM_ERR_TYPE_MISMATCH (The supplied value is incompatible with the type of the property.)
- CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other unspecified error occurred.)

5.4.2.20 GetQualifier

The GetQualifier operation retrieves a single qualifier declaration from the target namespace.

```
<qualifierDecl> GetQualifier (
   [IN] string QualifierName
)
```

The `QualifierName` input parameter identifies the qualifier with the declaration to be retrieved.

If GetQualifier is successful, the method returns the qualifier declaration for the named qualifier.

If GetQualifier is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE

- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_FOUND (The requested qualifier declaration does not exist.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.21 SetQualifier

The SetQualifier operation creates or updates a single qualifier declaration in the target namespace. If the qualifier declaration already exists, it is overwritten:

```
void SetQualifier (
   [IN] <qualifierDecl> QualifierDeclaration
)
```

The `QualifierDeclaration` input parameter defines the qualifier declaration to add to the namespace.

If SetQualifier is successful, the qualifier declaration is added to the target namespace. If a qualifier declaration with the same qualifier name already exists, the new declaration replaces it.

If SetQualifier is unsuccessful, this method returns one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.22 DeleteQualifier

The DeleteQualifier operation deletes a single qualifier declaration from the target namespace.

```
void DeleteQualifier (
   [IN] string QualifierName
)
```

The `QualifierName` input parameter identifies the qualifier with the declaration to be deleted.

If DeleteQualifier is successful, the specified qualifier declaration is deleted from the namespace.

If DeleteQualifier is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_FOUND (The requested qualifier declaration does not exist.)

- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.23 EnumerateQualifiers

The EnumerateQualifiers operation enumerates qualifier declarations from the target namespace.

```
<qualifierDecl>* EnumerateQualifiers (
)
```

If EnumerateQualifiers is successful, the method returns zero or more `<qualifierDecl>` items representing qualifier declarations.

If EnumerateQualifiers is unsuccessful, this method shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional method-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24 Pulled Enumeration Operations

This clause defines a set of operations that return CIM instances or instance paths in portions controlled by the WBEM client. These operations are called *pulled enumerations*. Usually, an enumeration session is established through an Open operation, and subsequent repeated executions of a Pull operation on the enumeration session are used to retrieve them. Optionally, the Open operation can also pull a first set of items.

Pulled enumeration operations consist of the following individual operations:

- Open operations open an enumeration of the following instances or instance paths:
  - OpenEnumerateInstances (instances of a class)
  - OpenEnumerateInstancePaths (instance paths of instances of a class)
  - OpenReferenceInstances (association instances referencing a target instance)
  - OpenReferenceInstancePaths (the instance paths of association instances referencing a target instance)
  - OpenAssociatorInstances (instances associated with a source instance)
  - OpenAssociatorInstancePaths (the instance paths of instances associated to a source instance)
  - OpenQueryInstances (the rows resulting from a query)
- Pull operations are for the following cases:
  - PullInstances (Instances are enumerated, and instance paths are either not available, for example as in for OpenQueryInstances, or not desired.)
  - PullInstancesWithPath (Instances with paths are enumerated.)
  - PullInstancePaths (Instance paths are enumerated.)

- Other operations are as follows:
  - CloseEnumeration (closes an open enumeration)
  - EnumerationCount (estimates the number of items in an open enumeration)

5.4.2.24.1 Behavioral Rules for Pulled Enumeration Operations

A central concept of pulled enumeration operations is the "enumeration session," which provides a context in which the operations perform their work and which determines the set of instances or instance paths to be returned. To process the operations of an enumeration session, some parameters of the Open operation need to be maintained as long as the enumeration session is open. In addition, some state data about where the enumeration session is with regard to instances or instance paths already returned must be maintained.

From a WBEM client perspective, an enumeration session is an enumeration context value. A successful Open operation establishes the enumeration session and returns an enumeration context value representing it. This value is used as an input/output parameter in subsequent Pull operations on that enumeration session. The enumeration context value shall uniquely identify the open enumeration session within the target CIM namespace of the Open operation that established the enumeration session. It is valid for a WBEM server to use NULL as an enumeration context value representing a closed enumeration session, but a WBEM client shall not rely on that.

Defining the enumeration context value in Pull operations as both an input parameter and an output parameter allows the WBEM server to change the enumeration context value during the execution of a pull operation. This ability to change allows different implementation approaches on the WBEM server side, which are transparent for the WBEM client. Example approaches are as follows:

- Maintain any state data describing the enumeration session internally in the WBEM server. The enumeration context value does not need to change in subsequent Pull operations. The WBEM server uses this value only to identify the internal state data for the open enumeration session. It does not use the value to store any state data. A variation of this approach is to hand back modified enumeration context values for additional WBEM server-side sequence checking.
- Maintain any state data describing the enumeration session only on the WBEM client side. All state data is stored in the enumeration context value, and the WBEM server does not maintain any state data about the enumeration session, essentially being completely stateless with regard to the enumeration session.
- A combination of the two previous approaches.

A WBEM server may support keeping enumeration sessions open across connection terminations and shutdowns of the server. Objects may be created, deleted, or modified concurrently with an enumeration session that involves these objects. Such changes may or may not be reflected in the enumeration set. Therefore, there is no guarantee to the WBEM client that the enumeration set represents a consistent snapshot of its instances at a point in time. However, the WBEM server should make a best effort attempt for the returned enumeration set to represent a consistent snapshot of its instances at a point in time. The order of instances in the enumeration set is undefined.

This document does not restrict the number of enumeration sessions that can be established or executed concurrently in the same WBEM server or client. This remains true even if the enumeration sets of such concurrently established enumeration sessions contain the same instances.

Except for CloseEnumeration, all operations on a particular enumeration session shall be executed sequentially. An enumeration session can be open or closed. It is considered open if operations using its enumeration context value as an input parameter can be executed successfully. It is opened by the successful completion of an Open operation and closed by one of the following events:

- Successful completion of a CloseEnumeration operation

- Successful completion of an open or pull operation with the `EndOfSequence` output parameter set to `true`
- Unsuccessful completion of a pull operation when `ContinueOnError` is not requested
- WBEM server-side decision to close the enumeration session based upon an operation timeout
- WBEM server-side decision to close an enumeration session during an operation on that enumeration session based upon exceeding server limits

A conformant WBEM server may support closure of enumeration sessions based upon exceeding server limits. Example situations for such a decision are:

- Pull operations with no objects requested that are repeated with a high frequency on the same enumeration session
- EnumerationCount operations repeated with a high frequency on the same enumeration session

A mechanism by which WBEM servers can declare support for closure of enumeration sessions based upon exceeding server limits is defined in 7.5. If a WBEM server supports such closure of enumeration sessions, it shall make the decision to close during an operation on that enumeration session. There is no way to indicate the reason for the closure if the decision is made elsewhere. If a WBEM server closes an enumeration session based upon exceeding server limits, it shall return failure on the operation on that enumeration session with the status code CIM_ERR_SERVER_LIMITS_EXCEEDED.

5.4.2.24.2 Common Parameters for the Open Operations

This clause defines commonly used parameters for the Open operations. The description of the individual Open operations references these parameters as appropriate. Note that not every Open operation uses every one of these common parameters:

- `EnumerationContext`
  - This output parameter is the enumeration context value representing the enumeration session. If the `EndOfSequence` is `true`, the `EnumerationContext` value may be NULL.
  - The representation of an enumeration context value uses a string type. In version 1.3 of this document, enumeration context values were represented using the ENUMERATIONCONTEXT XML element. The representation was changed to using a string type in version 1.4 of this document, because it had turned out that all known implementations had implemented the enumeration context value using a string type.
- `EndOfSequence`
  - This output parameter indicates to the WBEM client whether the enumeration session is exhausted. If `EndOfSequence` is `true` upon successful completion of an Open operation, no more instances are available and the WBEM server closes the enumeration session, releasing any allocated resources related to the enumeration session. If the enumeration set is empty, it is valid for a WBEM server to set `EndOfSequence` to `true`, even if `MaxObjectCount` is 0. In this case, the enumeration session is closed upon successful completion of the Open operation. If `EndOfSequence` is `false`, additional instances may be available and the WBEM server shall not close the enumeration session.
- `IncludeClassOrigin`
  - This input parameter is used only on Open operations that enumerate CIM instances. It controls whether information about the class origin of properties, references or methods is included in any enumerated CIM instances. If `IncludeClassOrigin` is `true`, the CLASSORIGIN attribute shall be present on all appropriate elements in each CIM instance returned by any subsequent PullInstance operations on this enumeration session. If

`IncludeClassOrigin` is `false`, any CLASSORIGIN attributes shall not be present in any enumerated instances.

- `FilterQueryLanguage` and `FilterQuery`
  - These input parameters specify a filter query that acts as an additional restricting filter on the set of enumerated instances.
  - WBEM servers shall support filter queries in pulled enumerations and shall support the DMTF Filter Query Language (FQL, see DSP0212) as a query language for such filter queries. WBEM servers may support additional query languages for pulled enumerations. A mechanism by which WBEM servers can declare the query languages they support for pulled enumerations is not defined in this document; it is anticipated that a CIM model based approach for declaring supported query languages is developed.

    Note that before version 1.4 of this document, support for filter queries in pulled enumerations was optional and no particular query language was required. As a consequence of this change, the status code CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED is no longer used in CIM-XML.
  - If `FilterQueryLanguage` is not NULL, it shall specify a query language and `FilterQuery` shall be a valid query in that query language.

    If the query language specified in `FilterQueryLanguage` is not supported by the WBEM server, it shall return an error with status code CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED.

    If the query language specified in `FilterQueryLanguage` is supported by the WBEM server, it shall process the filter query specified by the `FilterQuery` and `FilterQueryLanguage` parameters, and shall either restrict the set of enumerated instances as specified by the query language, or return an error.

    WBEM servers shall support the Filter Query Language (see DSP0212) as a query language for pulled enumerations. WBEM servers may support additional query languages for pulled enumerations.
  - The query specified in `FilterQuery` shall conform to the following:
    - If the query language supports specifying a set of classes the query applies to (for example, CQL in its FROM list), only the class named in the `ClassName` parameter shall be specified.
    - If the query language supports specifying a result list (for example, CQL in its SELECT list), a result list may be specified in the query, but the result list shall be ignored.
    - The query shall not define any ordering criteria or any grouping of objects.

    If the query does not satisfy these rules or if the query is invalid according to the definition of the query language, the WBEM server shall return an error with status code CIM_ERR_INVALID_QUERY. The Filter Query Language (see DSP0212) automatically satisfies these rules.
- `OperationTimeout`
  - This input parameter determines the minimum time the WBEM server shall maintain the open enumeration session after the last Open or Pull operation (unless the enumeration session is closed during the last operation). If the operation timeout is exceeded, the WBEM server may close the enumeration session at any time, releasing any resources allocated to the enumeration session.

- An `OperationTimeout` of 0 means that there is no operation timeout. That is, the enumeration session is never closed based on time.
- If `OperationTimeout` is NULL, the WBEM server shall choose an operation timeout.
- All other values for `OperationTimeout` specify the operation timeout in seconds.
- A WBEM server may restrict the set of allowable values for `OperationTimeout`. Specifically, the WBEM server may not allow 0 (no timeout). If the specified value is not an allowable value, the WBEM server shall return failure with the status code CIM_ERR_INVALID_OPERATION_TIMEOUT. A mechanism by which WBEM servers can declare the allowable values for `OperationTimeout` is defined in 7.5.

• `ContinueOnError`

- This input parameter, if true, requests a continuation on error, which is the ability to resume an enumeration session successfully after a Pull operation returns an error. A mechanism by which conformant WBEM servers can declare support for continuation on error is defined in 7.5.
- If a WBEM server does not support continuation on error and `ContinueOnError` is `true`, it shall return a failure with the status code CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED.
- If a WBEM server supports continuation on error and `ContinueOnError` is `true`, the enumeration session shall remain open when a Pull operation fails, and any subsequent successful Pull operations shall return the set of instances or instance paths that would have been returned if the failing Pull operations were successful. This behavior is subject to the consistency rules defined for pulled enumerations. If `ContinueOnError` is `false`, the enumeration session shall be closed when a Pull operation returns a failure.

• `MaxObjectCount`

- This input parameter defines the maximum number of instances or instance paths that this Open operation can return. Any uint32 number is valid, including 0. The WBEM server may deliver any number of instances or instance paths up to `MaxObjectCount` but shall not deliver more than `MaxObjectCount` elements. A conformant WBEM server implementation may choose to never return any instances or instance paths during an Open operation, regardless of the value of `MaxObjectCount`. Note that a WBEM client can use a `MaxObjectCount` value of 0 to specify that it does not want to retrieve any instances in the Open operation.

• Return Value (array of enumerated elements)

- The return value of a successful Open operation is an array of enumerated elements with a number of entries from 0 up to a maximum defined by `MaxObjectCount`. These entries meet the criteria defined in the Open operation. Note that returning no entries in the array does not imply that the enumeration session is exhausted. Only the `EndOfSequence` output parameter indicates whether the enumeration session is exhausted.

5.4.2.24.3 OpenEnumerateInstances

The OpenEnumerateInstances operation establishes and opens an enumeration session of the instances of a CIM class (including instances of its subclasses) in the target namespace. Optionally, it retrieves a first set of instances.

```
<instanceWithPath>* OpenEnumerateInstances (
   [OUT] string EnumerationContext,
   [OUT] Boolean EndOfSequence,
   [IN] <className> ClassName,
```

```
        [IN,OPTIONAL] boolean DeepInheritance = true,
        [IN,OPTIONAL] boolean IncludeClassOrigin = false,
        [IN,OPTIONAL,NULL] string PropertyList [] = NULL,
        [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
        [IN,OPTIONAL,NULL] string FilterQuery = NULL,
        [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
        [IN,OPTIONAL] Boolean ContinueOnError = false,
        [IN,OPTIONAL] uint32 MaxObjectCount = 0
    )
```

The OpenEnumerateInstances operation shall comply with the behavior defined in 5.4.2.24.1.

The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

The `ClassName` input parameter defines the class that is the basis for the enumeration. The enumeration set shall consist of all instances of that specified class, including any instances of any of its subclasses, in accordance with the polymorphic nature of CIM objects.

The `DeepInheritance` input parameter acts as a filter on the properties included in any enumerated CIM instances. If the `DeepInheritance` input parameter is `true`, all properties of each enumerated instance of the class shall be present (subject to constraints imposed by the other parameters), including any added by subclassing the specified class. If `DeepInheritance` is `false`, each enumerated instance includes only properties defined for the class specified by `ClassName`.

The `IncludeClassOrigin` input parameter is defined in 5.4.2.24.2.

The `PropertyList` input parameter acts as a filter on the properties in any enumerated CIM instances. If `PropertyList` is not NULL, the members of the array define zero or more property names of the specified class. This array may include inherited property names or property names explicitly defined in the specified class. However, it shall not include property names defined in subclasses of the specified class. Each enumerated instance shall not include any properties missing from this list. Note that `PropertyList` acts as an additional filter on the properties defined by the `DeepInheritance` input parameter. If `PropertyList` includes a property that is not in the set defined by `DeepInheritance`, the element for the property shall not be included. If `PropertyList` is an empty array, no properties are included in the enumerated instances. If `PropertyList` is NULL, no additional filtering is defined. If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise process the request normally. If `PropertyList` contains property names that are invalid for a target instance, the WBEM server shall ignore them for that instance but otherwise process the request normally.

The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

If OpenEnumerateInstances is successful, the return value shall be an array of `<instanceWithPath>` items representing enumerated instances as defined in 5.4.2.24.2.

The PullInstancesWithPath operation shall be used to pull instances for an enumeration session opened using OpenEnumerateInstances. If any other operation is used to pull instances, the WBEM server shall return failure with the status code CIM_ERR_FAILED.

If OpenEnumerateInstances is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_OPERATION_TIMEOUT
- CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not exist.)
- CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED
- CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is not recognized.)
- CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query language.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.4 OpenEnumerateInstancePaths

The OpenEnumerateInstancePaths operation establishes and opens an enumeration session of the instance paths of the instances of a CIM class (including instances of its subclasses) in the target namespace. Optionally, it retrieves a first set of instance paths:

```
<instancePath>* OpenEnumerateInstancePaths (
   [OUT] string EnumerationContext,
   [OUT] Boolean EndOfSequence,
   [IN] <className> ClassName,
   [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
   [IN,OPTIONAL,NULL] string FilterQuery = NULL,
   [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
   [IN,OPTIONAL] Boolean ContinueOnError = false,
   [IN,OPTIONAL] uint32 MaxObjectCount = 0
)
```

The OpenEnumerateInstancePaths operation shall comply with the behavior defined in 5.4.2.24.1.

The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

The `ClassName` input parameter defines the class that is the basis for the enumeration. The enumeration set shall consist of the instance paths of all instances of the specified class, including any instances of any of its subclasses, in accordance with the polymorphic nature of CIM objects.

The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

If OpenEnumerateInstancePaths is successful, the return value shall be an array of `<instancePath>` items representing enumerated instance paths as defined in 5.4.2.24.2.

The PullInstancePaths operation shall be used to pull instances for an enumeration session opened using OpenEnumerateInstancePaths. If any other operation is used to pull instances, the WBEM server shall return failure with the status code CIM_ERR_FAILED.

If OpenEnumerateInstancePaths is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_OPERATION_TIMEOUT
- CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not exist.)
- CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED
- CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is not recognized.)
- CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query language.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.5 OpenReferenceInstances

The OpenReferenceInstances operation establishes and opens the enumeration session of association instances that refer to a particular target CIM instance in the target namespace. Optionally, it retrieves a first set of instances:

```
<instanceWithPath>* OpenReferenceInstances (
   [OUT] string EnumerationContext,
   [OUT] Boolean EndOfSequence,
   [IN] <instanceName> InstanceName,
   [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
   [IN,OPTIONAL,NULL] string Role = NULL,
   [IN,OPTIONAL] boolean IncludeClassOrigin = false,
   [IN,OPTIONAL,NULL] string PropertyList [] = NULL,
   [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
   [IN,OPTIONAL,NULL] string FilterQuery = NULL,
   [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
   [IN,OPTIONAL] Boolean ContinueOnError = false,
```

```
        [IN,OPTIONAL] uint32 MaxObjectCount = 0
    )
```

The OpenReferenceInstances operation shall comply with the behavior defined in 5.4.2.24.1.

The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

The `InstanceName` input parameter specifies an instance name (model path) that identifies the target CIM instance with the referring association instances to be enumerated. Unless restricted by any of the filter parameters of this operation, the enumeration set shall consist of all association instances that reference the target instance.

The `ResultClass` input parameter, if not NULL, shall be a CIM class name. It acts as a filter on the enumerated set of instances by mandating that each enumerated instance shall be an instance of this class or one of its subclasses. The WBEM server shall not return an error if the `ResultClass` input parameter value is an invalid class name or if the class does not exist in the target namespace,

The `Role` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the enumerated set of instances by mandating that each enumerated instance shall refer to the target instance through a property with a name that matches the value of this parameter. The WBEM server shall not return an error if the `Role` input parameter value is an invalid property name or if the property does not exist,

The `IncludeClassOrigin` input parameter is defined in 5.4.2.24.2.

The `PropertyList` input parameter acts as a filter on the properties included in any enumerated CIM instances. If `PropertyList` is not `NULL`, the members of the array define zero or more property names. Each enumerated instance shall not include any properties missing from this list. If `PropertyList` is an empty array, no properties are included in each enumerated instance. If `PropertyList` is `NULL`, all properties are included in each enumerated instance, subject to the conditions expressed by the other parameters. If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise process the request normally. If `PropertyList` contains property names that are invalid for a target instance, the WBEM server shall ignore them for that instance but otherwise process the request normally. WBEM clients should not specify properties in `PropertyList` unless they specify a non-NULL value for the `ResultClass` parameter.

The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

If OpenReferenceInstances is successful, the return value shall be an array of `<instanceWithPath>` items representing enumerated instances as defined in 5.4.2.24.2.

The PullInstancesWithPath operation shall be used to pull instances for an enumeration session opened using OpenReferenceInstances. If any other operation is used to pull instances, the WBEM server shall return failure with the status code CIM_ERR_FAILED.

If OpenReferenceInstances is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element of and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED

- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_OPERATION_TIMEOUT
- CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise incorrect parameters)
- CIM_ERR_NOT_FOUND (The target instance was not found.)
- CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED
- CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is not recognized.)
- CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query language.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.6 OpenReferenceInstancePaths

The OpenReferenceInstancePaths operation establishes and opens an enumeration session of the instance paths of the association instances that refer to a particular target CIM instance in the target namespace. Optionally, it retrieves a first set of instance paths.

```
<instancePath>* OpenReferenceInstancePaths (
   [OUT] string EnumerationContext,
   [OUT] Boolean EndOfSequence,
   [IN] <instanceName> InstanceName,
   [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
   [IN,OPTIONAL,NULL] string Role = NULL,
   [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
   [IN,OPTIONAL,NULL] string FilterQuery = NULL,
   [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
   [IN,OPTIONAL] Boolean ContinueOnError = false,
   [IN,OPTIONAL] uint32 MaxObjectCount = 0
)
```

The OpenReferenceInstancePaths operation shall comply with the behavior defined in 5.4.2.24.1.

The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

The `InstanceName` input parameter specifies an instance name (model path) that identifies the target CIM instance with the referring association instances (respectively, their instance paths) to be enumerated. Unless restricted by any filter parameters of this operation, the enumeration set shall consist of the instance paths of all association instances that reference the target instance.

The `ResultClass` input parameter, if not NULL, shall be a CIM class name. It acts as a filter on the enumerated set of instance paths by mandating that each enumerated instance path shall identify an instance of this class or one of its subclasses. The WBEM server shall not return an error if the `ResultClass` input parameter value is an invalid class name or if the class does not exist in the target namespace.

The `Role` input parameter, if not NULL, shall be a property name. It acts as a filter on the enumerated set of instance paths by mandating that each enumerated instance path shall identify an instance that refers to the target instance through a property with a name that matches the value of this parameter. The WBEM server shall not return an error if the `Role` input parameter value is an invalid property name or if the property does not exist,

The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

If OpenReferenceInstancePaths is successful, the return value shall be an array of `<instancePath>` items representing enumerated instance paths as defined in 5.4.2.24.2.

The PullInstancePaths operation shall be used to pull instances for an enumeration session opened using OpenReferenceInstancePaths. If any other operation is used to pull instances, the WBEM server shall return failure with the status code CIM_ERR_FAILED.

If OpenReferenceInstancePaths is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_OPERATION_TIMEOUT
- CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_FOUND (The target instance was not found.)
- CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED
- CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is not recognized.)
- CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query language.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.7 OpenAssociatorInstances

The OpenAssociatorInstances operation establishes and opens an enumeration session of the instances associated with a particular source CIM instance in the target namespace. Optionally, it retrieves a first set of instances.

```
<instanceWithPath>* OpenAssociatorInstances (
   [OUT] string EnumerationContext,
   [OUT] Boolean EndOfSequence,
   [IN] <instanceName> InstanceName,
```

```
        [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
        [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
        [IN,OPTIONAL,NULL] string Role = NULL,
        [IN,OPTIONAL,NULL] string ResultRole = NULL,
        [IN,OPTIONAL] boolean IncludeClassOrigin = false,
        [IN,OPTIONAL,NULL] string PropertyList [] = NULL,
        [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
        [IN,OPTIONAL,NULL] string FilterQuery = NULL,
        [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
        [IN,OPTIONAL] Boolean ContinueOnError = false,
        [IN,OPTIONAL] uint32 MaxObjectCount = 0
    )
```

The OpenAssociatorInstances operation shall comply with the behavior defined in 5.4.2.24.1.

The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

The `InstanceName` input parameter specifies an instance name (model path) that identifies the source CIM instance with the associated instances to be enumerated. Unless restricted by any filter parameters of this operation, the enumeration set shall consist of all instances associated with the source instance.

The `AssocClass` input parameter, if not `NULL`, shall be a CIM association class name. It acts as a filter on the enumerated set of instances by mandating that each enumerated instance shall be associated with the source instance through an instance of this class or one of its subclasses. The WBEM server shall not return an error if the `AssocClass` input parameter value is an invalid class name or if the class does not exist in the target namespace.

The `ResultClass` input parameter, if not `NULL`, must be a CIM class name. It acts as a filter on the enumerated set of instances by mandating that each enumerated instance shall be an instance of this class or one of its subclasses. The WBEM server shall not return an error if the `ResultClass` input parameter value is an invalid class name or if the class does not exist in the target namespace.

The `Role` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the enumerated set of instances by mandating that each enumerated instance shall be associated with the source instance through an association in which the source instance plays the specified role. That is, the name of the property in the association class that refers to the source instance shall match the value of this parameter. The WBEM server shall not return an error if the `Role` input parameter value is an invalid property name or if the property does not exist.

The `ResultRole` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the enumerated set of instances by mandating that each enumerated instance shall be associated with the source instance through an association in which the enumerated instance plays the specified role. That is, the name of the property in the association class that refers to the enumerated instance shall match the value of this parameter. The WBEM server shall not return an error if the `ResultRole` input parameter value is an invalid property name or if the property does not exist.

The `IncludeClassOrigin` input parameter is defined in 5.4.2.24.2.

The `PropertyList` input parameter acts as a filter on the properties included in any enumerated CIM instances. If `PropertyList` is not NULL, the members of the array define zero or more property names. Each enumerated instance shall not include any properties missing from this list. If `PropertyList` is an empty array, no properties are included in each enumerated instance. If `PropertyList` is NULL, all properties are included in each enumerated instance, subject to the conditions expressed by the other parameters. If `PropertyList` contains duplicate property names, the WBEM server shall ignore them

but otherwise process the request normally. If `PropertyList` contains property names that are invalid for a target instance, the WBEM server shall ignore them for that instance but otherwise process the request normally. WBEM clients should not specify properties in `PropertyList` unless they specify a non-NULL value for the ResultClass parameter.

The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

If OpenAssociatorInstances is successful, the return value shall be an array of `<instanceWithPath>` items representing enumerated instances as defined in 5.4.2.24.2.

The PullInstancesWithPath operation shall be used to pull instances for an enumeration session opened using OpenAssociatorInstances. If any other operation is used to pull instances, the WBEM server shall return failure with the status code CIM_ERR_FAILED.

If OpenAssociatorInstances is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is given in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_OPERATION_TIMEOUT
- CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_FOUND (The source instance was not found.)
- CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED
- CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is not recognized.)
- CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query language.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.8 OpenAssociatorInstancePaths

The OpenAssociatorInstancePaths operation establishes and opens an enumeration session of the instance paths of the instances associated with a particular source CIM instance in the target namespace. Optionally, it retrieves a first set of instance paths.

```
<instancePath>* OpenAssociatorInstancePaths (
   [OUT] string EnumerationContext,
   [OUT] Boolean EndOfSequence,
   [IN] <instanceName> InstanceName,
   [IN,OPTIONAL,NULL] <className> AssocClass= NULL,
```

```
        [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
        [IN,OPTIONAL,NULL] string Role = NULL,
        [IN,OPTIONAL,NULL] string ResultRole = NULL,
        [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
        [IN,OPTIONAL,NULL] string FilterQuery = NULL,
        [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
        [IN,OPTIONAL] Boolean ContinueOnError = false,
        [IN,OPTIONAL] uint32 MaxObjectCount = 0
    )
```

This operation shall comply with the behavior defined in 5.4.2.24.1.

The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

The `InstanceName` input parameter specifies an instance name (model path) that identifies the source CIM instance with the associated instances (respectively, their instance paths) to be enumerated. Unless restricted by any filter parameters of this operation, the enumeration set shall consist of the instance paths of all instances associated with the source instance.

The `AssocClass` input parameter, if not `NULL`, shall be a CIM association class name. It acts as a filter on the enumerated set of instance paths by mandating that each instance path identify an instance that shall be associated with the source instance through an instance of this class or one of its subclasses. The WBEM server shall not return an error if the `AssocClass` input parameter value is an invalid class name or if the class does not exist in the target namespace.

The `ResultClass` input parameter, if not NULL, shall be a CIM class name. It acts as a filter on the enumerated set of instance paths by mandating that each instance path identify an instance that shall be an instance of this class or one of its subclasses. The WBEM server shall not return an error if the `ResultClass` input parameter value is an invalid class name or if the class does not exist in the target namespace.

The `Role` input parameter, if not NULL, shall be a property name. It acts as a filter on the enumerated set of instance paths by mandating that each instance path identify an instance that shall be associated with the source instance through an association in which the source instance plays the specified role. That is, the name of the property in the association class that refers to the source instance shall match the value of this parameter. The WBEM server shall not return an error if the `Role` input parameter value is an invalid property name or if the property does not exist.

The `ResultRole` input parameter, if not NULL, shall be a property name. It acts as a filter on the enumerated set of instance paths by mandating that each instance path identify an instance that shall be associated with the source instance through an association in which the instance identified by the enumerated instance path plays the specified role. That is, the name of the property in the association class that refers to the instance identified by the enumerated instance path shall match the value of this parameter. The WBEM server shall not return an error if the `ResultRole` input parameter value is an invalid property name or if the property does not exist.

The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

If OpenAssociatorInstancePaths is successful, the return value shall be an array of `<instancePath>` items representing enumerated instance paths as defined in 5.4.2.24.2.

The PullInstancePaths operation shall be used to pull instances for an enumeration session opened using OpenAssociatorInstancePaths. If any other operation is used to pull instances, the WBEM server shall return failure with the status code CIM_ERR_FAILED.

If OpenAssociatorInstancePaths is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_OPERATION_TIMEOUT
- CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_FOUND (The source instance was not found.)
- CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED
- CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is not recognized.)
- CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter language.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.9 Common Parameters for the Pull Operations

This clause defines commonly used parameters for the Pull operations. The description of the individual Pull operations references these parameters as appropriate. Note that not every Pull operation uses every one of these common parameters.

- `EnumerationContext`
  - This parameter is the enumeration context value representing the enumeration session to be used.
  - The representation of an enumeration context value uses a string type. In version 1.3 of this document, enumeration context values were represented using the ENUMERATIONCONTEXT XML element. The representation was changed to using a string type in version 1.4 of this document, because it had turned out that all known implementations had implemented the enumeration context value using a string type.
  - When the Pull operation is invoked, the enumeration session represented by the `EnumerationContext` input parameter shall be open. The first enumeration session shall use one of the Open operations with a type of enumerated object that matches the Pull operation. For the first Pull operation on an enumeration session, the value of the `EnumerationContext` input parameter shall be the enumeration context value returned by a successful Open operation. For subsequent Pull operations on that enumeration session, the value of the `EnumerationContext` input parameter shall be the value of the

EnumerationContext output parameter returned by the previous Pull operation on the same enumeration session.

  - After the Pull operation is completed, the enumeration session represented by the EnumerationContext output parameter shall be open or closed.

- EndOfSequence

  - This output parameter indicates to the WBEM client whether the enumeration session is exhausted. If EndOfSequence is true upon successful completion of a Pull operation, no more instances or instance paths are available and the WBEM server shall close the enumeration session, releasing any allocated resources related to the session. If EndOfSequence is false, additional instances or instance paths may be available, and the WBEM server shall not close the session.

- MaxObjectCount

  - This input parameter defines the maximum number of instances or instance paths that may be returned by this Pull operation. Any uint32 number is valid, including 0. The WBEM server may deliver any number of instances or instance paths up to MaxObjectCount but shall not deliver more than MaxObjectCount. The WBEM client may use a MaxObjectCount value of 0 to restart the operation timeout for the enumeration session when it does not need to not retrieve any instances or instance paths.

- Return Value (array of enumerated elements)

  - The return value of a Pull operation upon successful completion is an array of enumerated instances or instance paths with a number of entries from 0 up to a maximum defined by MaxObjectCount. These entries meet the criteria defined in the Open operation that established this enumeration session. Note that returning no entries in the array does not imply that the enumeration session is exhausted. Only the EndOfSequence output parameter indicates whether the enumeration session is exhausted.

5.4.2.24.10 PullInstancesWithPath

The PullInstancesWithPath operation retrieves instances including their instance paths from an open enumeration session represented by an enumeration context value:

```
<instanceWithPath>* PullInstancesWithPath (
   [IN,OUT] string EnumerationContext,
   [OUT] Boolean EndOfSequence,
   [IN] uint32 MaxObjectCount
)
```

The PullInstancesWithPath operation shall comply with the behavior defined in 5.4.2.24.1.

The EnumerationContext input/output parameter is defined in 5.4.2.24.9. The enumeration session shall be established using one of the OpenEnumerateInstances, OpenReferenceInstances, or OpenAssociatorInstances operations.

The EndOfSequence output parameter is defined in 5.4.2.24.9.

The MaxObjectCount input parameter is defined in 5.4.2.24.9.

If PullInstancesWithPath is successful, the return value shall be an array of <instanceWithPath> items representing enumerated instances including their instance paths as defined in 5.4.2.24.9.

If PullInstancesWithPath is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_ENUMERATION_CONTEXT
- CIM_ERR_PULL_HAS_BEEN_ABANDONED
- CIM_ERR_SERVER_LIMITS_EXCEEDED
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.11PullInstancePaths

The PullInstancePaths operation retrieves instance paths from an open enumeration session represented by an enumeration context value:

```
<instancePath>* PullInstancePaths (
   [IN,OUT] string EnumerationContext,
   [OUT] Boolean EndOfSequence,
   [IN] uint32 MaxObjectCount
)
```

The PullInstancePaths operation shall comply with the behavior defined in 5.4.2.24.1.

The `EnumerationContext` input/output parameter is defined in 5.4.2.24.9. The enumeration session shall have been established using one of the OpenEnumerateInstancePaths, OpenReferenceInstancePaths, or OpenAssociatorInstancePaths operations.

The `EndOfSequence` output parameter is defined in 5.4.2.24.9.

The `MaxObjectCount` input parameter is defined in 5.4.2.24.9.

If PullInstancePaths is successful, the return value shall be an array of `<instancePath>` items representing enumerated instance paths as defined in 5.4.2.24.9.

If PullInstancePaths is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_ENUMERATION_CONTEXT
- CIM_ERR_SERVER_LIMITS_EXCEEDED
- CIM_ERR_PULL_HAS_BEEN_ABANDONED

- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.12CloseEnumeration

The CloseEnumeration operation closes an open enumeration session, performing an early termination of an enumeration sequence:

```
void CloseEnumeration (
   [IN] string EnumerationContext
)
```

The `EnumerationContext` parameter is the value representing the enumeration session to be closed. The enumeration session shall be open and shall be established using one of the Open operations. This implies that this operation is not to close an enumeration sequence already indicated by `EndOfSequence` because the sequence has already been closed. The value of the `EnumerationContext` parameter shall be the value of the `EnumerationContext` output parameter returned by the previous Pull operation on the enumeration session to be closed.

If CloseEnumeration is successful, the WBEM server shall close the enumeration session represented by `EnumerationContext`, releasing any allocated resources. Any subsequent use of the `EnumerationContext` value is unsuccessful.

CloseEnumeration may be executed concurrently with a Pull operation or an EnumerationCount operation on the same enumeration session. If a WBEM server receives a CloseEnumeration operation request while it is processing a Pull operation on the same enumeration session, the WBEM server shall attempt to abandon that Pull operation. If the Pull operation can be abandoned, it shall return a failure with the status code CIM_ERR_PULL_HAS_BEEN_ABANDONED and the CloseEnumeration operation shall return success. If the Pull operation cannot be abandoned, it shall proceed as if the CloseEnumeration operation has not been issued, and the CloseEnumeration operation shall return a failure with the status code CIM_ERR_PULL_CANNOT_BE_ABANDONED.

If CloseEnumeration is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_ENUMERATION_CONTEXT
- CIM_ERR_PULL_CANNOT_BE_ABANDONED
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.13EnumerationCount

The EnumerationCount operation provides an estimated count of the total number of objects in an open enumeration session represented by an EnumerationContext:

```
uint64 EnumerationCount (
   [IN] string EnumerationContext
)
```

The `EnumerationContext` parameter identifies the enumeration session for the EnumerationCount operation. It shall be established using any of the Open operations and shall be open at the time of the CloseEnumeration request. A conformant WBEM server may support this operation. A WBEM server that does not support this operation should respond with the CIM_ERR_NOT_SUPPORTED status.

If EnumerationCount is successful, the operation returns an approximate count of the number of objects in the enumeration session. This is the number of items remaining to be sent with subsequent Pull operations. Thus, executing this operation immediately after the open may provide an approximate estimate of the total number of objects to be returned in the enumeration set. The returned count is only an estimate of the number of objects to be pulled in the enumeration sequence. This mechanism is intended to assist WBEM clients in determining the overall size of an enumeration set and the number of objects remaining in the enumeration session. It should not be used instead of the `EndOfSequence` parameter to determine the end of an enumeration sequence.

If the WBEM server cannot or will not return an estimate of the number of objects to be returned for the enumeration context, it may return success and the NULL value.

If EnumerationCount is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_ENUMERATION_CONTEXT
- CIM_ERR_SERVER_LIMITS_EXCEEDED
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.14OpenQueryInstances

The OpenQueryInstances operation establishes and opens an enumeration session of the instances of a CIM class (including instances of its subclasses) in the target namespace. Optionally, it retrieves a first set of instances:

```
<instance>* OpenQueryInstances (
   [IN] string FilterQuery,
   [IN] string FilterQueryLanguage,
   [IN,OPTIONAL] Boolean ReturnQueryResultClass = false,
   [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
   [IN,OPTIONAL] Boolean ContinueOnError = false,
   [IN,OPTIONAL] uint32 MaxObjectCount = 0,
   [OUT, OPTIONAL, NULL] <class> QueryResultClass,
   [OUT] string EnumerationContext,
   [OUT] Boolean EndOfSequence
)
```

The OpenQueryInstances shall comply with the behavior defined in 5.4.2.24.1.

The `FilterQuery` and `FilterQueryLanguage` input parameters specify the set of enumerated instances.

`FilterQueryLanguage` shall specify a query language and the value of `FilterQuery` shall be a valid query in that query language. This document defines neither the query language nor the format of the query. It is anticipated that query languages will be submitted to the DMTF as separate proposals. A mechanism by which WBEM servers can declare the query languages they support for filtering in Pulled enumerations (if any) is defined in 7.5.

The `ReturnQueryResultClass` input parameter controls whether a class definition is returned in `QueryResultClass`. If it is set to `false`, `QueryResultClass` shall be set to NULL on output. If it is set to `true`, the value of the `QueryResultClass` on output shall be a class definition that defines the properties (columns) of each row of the query result.

The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

The `QueryResultClass` output parameter shall be set to NULL if the `ReturnQueryResultClass` input parameter is set to `false`. Otherwise, it shall return a class definition where each property of the class corresponds to one entry of the query select list. The class definition corresponds to one row of the query result. The class name of this returned class shall be "CIM_QueryResult." This class definition is valid only in the context of this enumeration.

The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

If OpenQueryInstances is successful, the return value shall be an array of `<instance>` items representing enumerated instances as defined in 5.4.2.24.2. Such instances are available only in the context of the enumeration and do not return an instance path. The PullInstancesWithPath operation may not be used to continue an enumeration started by the OpenQueryInstances operation.

The PullInstances operation shall be used to pull instances for an enumeration session opened using If OpenQueryInstances. If any other operation is used to pull instances, the WBEM server shall return failure with the status code CIM_ERR_FAILED.

If OpenQueryInstances is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_OPERATION_TIMEOUT
- CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is not recognized.)
- CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query language.)

- CIM_ERR_QUERY_FEATURE_NOT_SUPPORTED (The query requires support for features that are not supported.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.15PullInstances

The PullInstances operation retrieves instances from an OpenQueryInstances session represented by an enumeration context value:

```
<instance>* PullInstances (
   [IN,OUT] string EnumerationContext,
   [OUT] Boolean EndOfSequence,
   [IN] uint32 MaxObjectCount
)
```

The PullInstances operation shall comply with the behavior defined in 5.4.2.24.1.

The `EnumerationContext` input/output parameter is defined in 5.4.2.24.9. The enumeration session shall be established using the OpenQueryInstances operation.

The `EndOfSequence` output parameter is defined in 5.4.2.24.9.

The `MaxObjectCount` input parameter is defined in 5.4.2.24.9.

If PullInstances is successful, the return value shall be an array of `<instance>` items representing enumerated instances as defined in 5.4.2.24.9.

If PullInstances is unsuccessful, this operation shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_SERVER_IS_SHUTTING_DOWN
- CIM_ERR_NOT_SUPPORTED
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_INVALID_ENUMERATION_CONTEXT
- CIM_ERR_SERVER_LIMITS_EXCEEDED
- CIM_ERR_PULL_HAS_BEEN_ABANDONED
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.3 Namespace Manipulation Using the CIM_Namespace Class

No intrinsic methods are defined specifically to manipulate namespaces. Namespaces shall be manipulated using intrinsic methods on the CIM_Namespace class.

5.4.3.1 Namespace Creation

A namespace is created by calling the intrinsic method CreateInstance for the CIM_Namespace class. A value is specified for the new instance parameter that defines a valid instance of the CIM_Namespace class and that has a name property that is the desired name of the new namespace.

The proposed definition shall be a correct namespace definition according to DSP0004. Despite the naming conventions used in the CIM specifications (use of / in namespaces such as root/CIMV2 and root/CIMV2/test), there is no hierarchy implied among different namespaces. Each namespace is independent of all others. The namespaces are to be considered flat, and there is no defined behavior for navigating namespaces.

In creating the new namespace, the WBEM server shall conform to the following rules:

- The namespace defined by name property shall not already exist in the WBEM server.
- The <LOCALNAMESPACEPATH> defined for the operation defines the namespace in which the CIM_Namespace instance associated with this new namespace is created.

It is recommended that instances of CIM_Namespace be created in root unless there is a specific reason to define them in another namespace. The inclusion of a CIM_Namespace instance within a namespace other than root is allowed.

In addition to creating instances of CIM_Namespace, compliant implementations shall also create an instance of the association class CIM_NamespaceInManager defining the linking of the namespace created to the current CIM_ObjectManager.

If CreateInstance is successful, the WBEM server creates the specified namespace. In addition, the WBEM server shall return information about the namespace as an instance of the class CIM_Namespace and of returning instances of the association class CIM_NamespaceInManager for each CIM_Namespace instance created.

5.4.3.2 Namespace Deletion

If the WBEM server supports the CIM_Namespace class, all valid namespaces shall be represented by an instance of the CIM_Namespace class. A namespace is deleted using the intrinsic method DeleteInstance to delete the instance of the class CIM_Namespace that represents the namespace. The namespace to be deleted shall exist.

If DeleteInstance is successful, the WBEM server shall remove the specified CIM_Namespace instance.

If DeleteInstance is unsuccessful, one of the status codes defined for the DeleteInstance operation shall be returned. A WBEM server may return CIM_ERR_FAILED if a non-empty namespace cannot successfully be deleted.

5.4.3.3 Manipulation and Query of Namespace Information

The query of namespaces is provided through the following means:

- Query of the CIM_Namespace class on an individual namespace
- Use of the CIM_NamespaceInManager association to link the target CIM_ObjectManager and the instances of CIM_Namespace representing all namespaces defined in the target CIM_ObjectManager

5.4.3.4 Use of the __Namespace Pseudo Class (DEPRECATED)

In previous versions of this document, namespaces were manipulated through the pseudo class __Namespace as follows:

No intrinsic methods are specifically defined for manipulating CIM namespaces. However, modeling a CIM namespace using class __Namespace, together with the requirement that the root namespace be supported by all WBEM servers, implies that all namespace operations can be supported.

For example, all child namespaces of a particular namespace are enumerated by calling the intrinsic method EnumerateInstanceNames against the parent namespace, specifying a value for the ClassName

parameter of __Namespace. A child namespace is created by calling the intrinsic method CreateInstance against the parent namespace, specifying a value for the NewInstance parameter that defines a valid instance of the class __Namespace and that has a name property that is the desired name of the new namespace.

DEPRECATION NOTE: The use of the __Namespace class is DEPRECATED. In its place, use the CIM_Namespace class.

5.4.4 Functional Profiles

To establish conformance, this clause partitions the intrinsic methods into functional groups.

Support for a particular group does *not* guarantee that all invocations of a method in that group will succeed. Rather, the exclusion of a group is a declaration that any attempt to call a method in that group always returns CIM_ERR_NOT_SUPPORTED.

Mechanisms by which a WBEM server may declare the functional groups that it supports are defined in 7.5.

To limit the number of different profiles that a WBEM server may support, each functional group has a dependency on another group (with the exception of the Basic Read functional group). If functional group G_1 has a dependency on functional group G_2, then a WBEM server that supports G_1 shall also support G_2.

The dependency relation is transitive, so if G_1 depends on G_2, and G_2 depends on G_3, then G_1 depends on G_3. It is also anti-symmetric, so if G_1 depends on G_2, then G_2 cannot depend on G_1.

Using these rules, Table 3 defines a rooted-directed tree of dependencies with the Basic Read dependency representing the root node.

For example, a WBEM server that supports the Schema Manipulation functional group shall also support the Instance Manipulation, Basic Write, and Basic Read.

A WBEM server shall support the Basic Read functional group.

2603 **Table 3 – Root-Directed Tree of Functional Profile Dependencies**

Functional Group	Dependency	Methods
Basic Read	none	GetClass EnumerateClasses EnumerateClassNames GetInstance EnumerateInstances EnumerateInstanceNames GetProperty (DEPRECATED)
Pulled Read	Basic Read	OpenEnumerateInstances OpenEnumerateInstancePaths OpenReferenceInstances OpenReferenceInstancePaths OpenAssociatorInstances OpenAssociatorInstancePaths PullInstancesWithPath PullInstancePaths CloseEnumeration
PulledReadCount	Pulled Read	EnumerationCount
Pulled Query Execution	Pulled Read	OpenQueryInstances PullInstances
Basic Write	Basic Read	SetProperty (DEPRECATED)
Schema Manipulation	Instance Manipulation	CreateClass ModifyClass DeleteClass
Instance Manipulation	Basic Write	CreateInstance ModifyInstance DeleteInstance
Association Traversal	Basic Read	Associators AssociatorNames References ReferenceNames
Query Execution	Basic Read	ExecQuery
Qualifier Declaration	Schema Manipulation	GetQualifier SetQualifier DeleteQualifier EnumerateQualifiers

2604 ### 5.4.5 Extrinsic Method Invocation

2605 Any WBEM server is assumed to support extrinsic methods, which are defined by the schema supported
2606 by the WBEM server. If a WBEM server does not support extrinsic method invocations, it shall return the
2607 error code CIM_ERR_NOT_SUPPORTED to any request to execute an extrinsic method (subject to the
2608 considerations described in the rest of this clause). This allows a WBEM client to determine that all
2609 attempts to execute extrinsic methods will fail.

If the WBEM server cannot invoke extrinsic methods, it shall return one of the following status codes, where the error returned is the first applicable error in the list, starting with the first element and working down. Any additional specific interpretation of the error is enclosed in parentheses.

- CIM_ERR_ACCESS_DENIED
- CIM_ERR_NOT_SUPPORTED (The WBEM server does not support extrinsic method invocations.)
- CIM_ERR_INVALID_NAMESPACE
- CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
- CIM_ERR_NOT_FOUND (The target CIM class or instance does not exist in the specified namespace.)
- CIM_ERR_METHOD_NOT_FOUND
- CIM_ERR_METHOD_NOT_AVAILABLE (The WBEM server is unable to honor the invocation request.)
- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.5 CIM Export Syntax and Semantics

This clause focuses on export methods and their invocation, as well as on functional profiles.

5.5.1 Export Method Invocations

All CIM-XML export message requests defined for the CIM-to-HTTP mapping are invocations of one or more export methods. Export methods do not operate against CIM namespaces.

An export method call is represented in XML by the <EXPMETHODCALL> element, and the response to that call is represented by the <EXPMETHODRESPONSE> element.

An input parameter has an IN qualifier with value `true` in the method definition. An output parameter has an OUT qualifier with value `true` in the method definition. A parameter may be both an input parameter and an output parameter.

The <EXPMETHODCALL> element names the method to be invoked and supplies any input parameters to the export method call:

- Each input parameter shall be named using the name assigned in the method definition.
- Input parameters may be supplied in any order.
- Each input parameter of the method, and no others, shall be present in the call unless it is optional.

The <EXPMETHODRESPONSE> element defines either an <ERROR> or a (possibly optional) return value and output parameters, which are decorated with the OUT qualifier in the method definition. In the latter case, the following rules apply:

- Each output parameter shall be named using the name assigned in the method definition.
- Output parameters may be supplied in any order.
- Each output parameter of the method, and no others, shall be present in the response, unless it is optional.

The method invocation process may be thought of as a two-part process:

- Binding the input parameter values specified as child elements of the <EXPMETHODCALL> element to the input parameters of the method.
- Attempting to execute the method using the bound input parameters, with one of the following results:
  - If the attempt to call the method is successful, the return value and output parameters are bound to the child elements of the <EXPMETHODRESPONSE> element.
  - If the attempt to call the method is unsuccessful, an error code and (optional) human-readable description of that code is bound to the <EXPMETHODRESPONSE> element.

5.5.1.1 Simple Export

A simple export requires the invocation of a single export method. A simple export request is represented by a <SIMPLEEXPREQ> element, and a simple export response is represented by a <SIMPLEEXPRSP> element.

A <SIMPLEEXPREQ> shall contain a <EXPMETHODCALL> element.

5.5.1.2 Multiple Export

A multiple export requires the invocation of more than one export method. A multiple export request is represented by a <MULTIEXPREQ> element, and a multiple export response is represented by a <MULTIEXPRSP> element.

A <MULTIEXPREQ> (or its respective <MULTIEXPRSP>) element is a sequence of two or more <SIMPLEEXPREQ> (or its respective <SIMPLEEXPRSP>) elements.

A <MULTIEXPRSP> element shall contain a <SIMPLEEXPRSP> element for every <SIMPLEEXPREQ> element in the corresponding multiple export response. These <SIMPLEEXPRSP> elements shall be in the same order as their <SIMPLEEXPREQ> counterparts. The first <SIMPLEEXPRSP> in the response corresponds to the first <SIMPLEEXPREQ> in the request, and so forth.

Multiple exports conveniently batch the delivery of multiple export method invocations into a single HTTP message, reducing the number of roundtrips between a WBEM client and a WBEM listener and allowing the WBEM listener to make certain internal optimizations. Note that multiple exports do not confer any transactional capabilities in processing the request. For example, the WBEM listener does not have to guarantee that the constituent export method calls either all failed or all succeeded. The WBEM listener must only make a "best effort" to process the operation. However, WBEM listeners shall finish processing each method invocation in a batched message before executing the next method invocation in the batch. Clients shall recognize that the order of method calls within a batched message is significant.

Not all WBEM listeners support multiple exports. If a WBEM listener does not support multiple exports, it shall return the status code CIM_ERR_NOT_SUPPORTED.

5.5.1.3 Status Codes

This clause defines the status codes and detailed error information that a conforming WBEM listener may return.

The value of an <ERROR> child element within a <EXPMETHODRESPONSE> element includes the following parts:

- mandatory status code
- optional human-readable description of the status code
- zero or more CIM_Error instances

The symbolic names defined in Table 4 do not appear on the wire. They are used here solely for convenient reference to an error in other parts of this document. Not all methods are expected to return all these status codes.

In addition to returning a status code, a conforming WBEM listener may return zero or more <INSTANCE> child elements as part of an <ERROR> element. Each <INSTANCE> child element shall be an instance of CIM_Error, and the value of CIMStatusCode shall comply with the definition of expected error codes for the CIM-XML export request. A WBEM client may ignore any <INSTANCE> child elements.

Table 4 – Symbolic Names for Referencing Error Codes

Symbolic Name	Code	Definition
CIM_ERR_FAILED	1	A general error occurred that is not covered by a more specific error code.
CIM_ERR_ACCESS_DENIED	2	Access was not available to the client.
CIM_ERR_NOT_SUPPORTED	7	The requested operation is not supported.
CIM_ERR_TYPE_MISMATCH	13	The value supplied is incompatible with the type.

5.5.2 Export Methods

This clause describes the methods that can be defined within a CIM-XML export message. These methods operate only on an external data representation of a CIM entity, namespace, or element. Specifically, export methods do not operate on CIM namespaces or elements. The export method defined in this document is Export an Indication.

The notation used in the following subclauses to define the signatures of the export methods is a pseudo-MOF notation that extends the standard MOF BNF (DSP0004) for describing CIM export methods with a number of pseudo parameter types. The pseudo parameter types are enclosed in angle brackets (< >).

This notation allows parameters to be decorated with pseudo-qualifiers (IN, OPTIONAL, and NULL) to define their invocation semantics. Note that these qualifiers are for description purposes only within the scope of this document. In particular, a WBEM client shall not specify them in export method invocations.

This notation uses the IN qualifier for input parameters.

A WBEM client may omit an optional parameter if the required value is the specified default by not specifying an <EXPPARAMVALUE> element for the parameter. It shall not omit a parameter that is not optional.

The NULL qualifier indicates parameters with values that may be specified as NULL in an export method call. A NULL (unassigned) value for a parameter is specified by an <EXPPARAMVALUE> element with no child element. The WBEM client shall specify a value for parameters without the NULL qualifier by including a suitable child element for the <EXPPARAMVALUE> element.

All parameters shall be uniquely named and shall correspond to a valid parameter name for that method as described by this document. The order of the parameters is not significant.

The non-NULL values of export method parameters or return values that are modeled as standard CIM types (such as string and Boolean, or arrays thereof) are represented as follows:

- Simple values shall be represented by the <VALUE> child element in an <EXPPARAMVALUE> element (for export method parameters) or in an <IRETURNVALUE> element (for export method return values).

- Array values shall be represented by the <VALUE.ARRAY> child element in an <EXPPARAMVALUE> element (for export method parameters) or in an <IRETURNVALUE> element (for export method return values).

Table 5 shows how each pseudo-type used by the export methods shall be mapped to an XML element described in DSP0201 in the context of both a parameter value (child element of <EXPPARAMVALUE>) and a return value (child element of <IRETURNVALUE>).

2732 **Table 5 – Mapping of Export Method Pseudo-Types to XML Elements**

Type	XML Element
<object>	(VALUE.OBJECT|VALUE.OBJECTWITHLOCALPATH|VALUE.OBJECTWITHPATH)
<class>	CLASS
<instance>	INSTANCE
<className>	CLASSNAME
<namedInstance>	VALUE.NAMEDINSTANCE
<instanceName>	INSTANCENAME
<objectWithPath>	VALUE.OBJECTWITHPATH
<objectName>	(CLASSNAME|INSTANCENAME)
<propertyValue>	(VALUE|VALUE.ARRAY|VALUE.REFERENCE)
<qualifierDecl>	QUALIFIER.DECLARATION

2733 #### 5.5.2.1 ExportIndication

2734 The ExportIndication operation exports a single CIM indication to the destination WBEM listener:

```
2735    void ExportIndication (
2736       [IN] <instance> NewIndication
2737    )
```

2738 The `NewIndication` input parameter defines the indication to be exported. The proposed definition
2739 should be a correct instance definition for the underlying CIM indication class according to the CIM
2740 specification.

2741 If ExportIndication is unsuccessful, this method shall return one of the following status codes, where the
2742 error returned is the first applicable error in the list, starting with the first element and working down. Any
2743 additional method-specific interpretation of the error is enclosed in parentheses.

2744 - CIM_ERR_ACCESS_DENIED
2745 - CIM_ERR_NOT_SUPPORTED
2746 - CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2747 incorrect parameters)
2748 - CIM_ERR_INVALID_CLASS (The CIM class of which this is to be a new instance does not
2749 exist.)
2750 **DEPRECATED:** The use of CIM_ERR_INVALID_CLASS has been deprecated in version 1.4 of
2751 this document because a WBEM listener has no notion about existing classes. Listeners should
2752 not use this status code anymore, and WBEM servers receiving this status code should treat it
2753 like CIM_ERR_FAILED.
2754 - CIM_ERR_FAILED (Some other unspecified error occurred.)

2755 ### 5.5.3 Functional Profiles

2756 This clause partitions the export methods into functional groups to establish conformance. See Table 6.

2757 Support for a particular group does not guarantee that all invocations of an export method in that group
2758 will succeed. Rather, the exclusion of a group is a declaration that any attempt to call an export method in
2759 that group always returns CIM_ERR_NOT_SUPPORTED.

The dependency relation is transitive, so if group G_1 depends on G_2, and G_2 depends on G_3, then G_1 depends on G_3. It is also anti-symmetric, so if G_1 depends on G_2, then G_2 cannot depend on G_1.

Table 6 – Functional Groups of Export Methods

Functional Group	Dependency	Method
Indication	None	ExportIndication

6 Encapsulation of CIM-XML Messages

This clause describes how to use CIM-XML messages in HTTP. CIM-XML message requests may be used with or without the HTTP Extension Framework.

Although CIM-XML messages can be used in combination with a variety of HTTP request methods, this document defines CIM-XML messages only within HTTP POST requests. (M-POST may be used in place of POST. For details on how to use CIM-XML messages with the HTTP Extension Framework, see 6.2.)

All CIM-XML message responses are carried in the corresponding HTTP response. In the remaining discussion, the following terms are used as convenient shorthand for the definitions provided here:

- *CIM-XML operation request*. An HTTP POST request message with an XML entity body that defines an operation request message.
- *CIM-XML operation response*. An HTTP response message, issued in response to a CIM-XML operation request, with an entity body that defines an operation response message.
- *CIM-XML export request*. An HTTP POST request message with an XML entity body that defines an export request message.
- *CIM-XML export response*. An HTTP response message, issued in response to a CIM-XML export message request, with an entity body that defines an export response message.
- *CIM-XML message request*. An HTTP POST request message with an XML entity body that defines either an operation request message or an export request message.
- *CIM-XML message response*. An HTTP response message, issued in response to a CIM-XML message request, with an entity body that defines either an operation response message or an export response message.

Note that an HTTP response to a CIM request is not always a CIM response. For example, a "505 HTTP Version Not Supported" response is not a CIM response.

6.1 WBEM clients, WBEM servers, and WBEM listeners

A *CIM product* is any product that can supply and/or consume management information using the CIM schema. In particular, WBEM clients, WBEM servers, and WBEM listeners are examples of CIM products:

- A *WBEM client* issues CIM-XML operation requests and receives and processes CIM-XML operation responses.
- A *WBEM server* receives and processes CIM-XML operation requests and issues CIM-XML operation responses. A WBEM server also issues CIM-XML export requests and receives and processes CIM-XML export responses.
- A *WBEM listener* is a server that receives and processes CIM-XML export requests and issues CIM-XML export responses.

Throughout this document, the terms WBEM client, WBEM server, WBEM listener, and CIM product are used as convenient shorthand to refer to the subset of CIM products that conform to this document.

Note that "WBEM client" (server, listener) was used for the term "WBEM client" (server, listener) before version 1.4 of this document.

6.2 Use of M-POST

A WBEM client attempting to invoke a CIM-XML message using the HTTP Extension Framework method "M-POST" shall follow these steps:

- If the M-POST invocation fails with an HTTP status of "501 Not Implemented" or "510 Not Extended," the client should retry the request using the HTTP method "POST" with the appropriate modifications (described in 6.2.2).
- If the M-POST invocation fails with an HTTP status of "405 Method Not Allowed," the client should fail the request.
- For all other status codes, the client shall act in accordance with standard HTTP (see 7.1).

This extended invocation mechanism gives Internet proxies and firewalls greater filtering control and administrative flexibility over CIM-XML message invocations.

If a client receives a 501 or 510 status in response to an M-POST request, in subsequent invocations to the same HTTP server, the client may omit the attempt at M-POST invocations for a suitable period. This omission avoids the need for an extra round trip on each and every method invocation. The details of the caching strategy employed by the client are outside the scope of this document.

6.2.1 Use of the Ext Header

If a WBEM server or WBEM listener receives a valid M-POST request and has fulfilled all mandatory extension header declarations in the request, it shall include in the response the "Ext" header defined by RFC2774. This included header shall be protected by the appropriate Cache-Control directive.

6.2.2 Naming of Extension Headers

In M-POST request messages (and their responses), CIM extension headers shall be declared using the name space prefix allotted by the "Man" extension header (in accordance with RFC2774) that refers to the name space "http://www.dmtf.org/cim/mapping/http/v1.0". The full format of the "Man" header declaration for this document is:

```
Man           = "Man" ":" "http://www.dmtf.org/cim/mapping/http/v1.0"
                ";" "ns" "=" header-prefix

header-prefix = 2*DIGIT
```

This header-prefix should be generated at random on a per-HTTP message basis, and should not necessarily be a specific number.

In accordance with RFC2774, all POST request messages (and their responses) shall not include such a mandatory extension declaration. In POST request messages (and their responses), name space prefixes shall not be used.

EXAMPLE 1:

Using M-POST:

```
M-POST /cimom HTTP/1.1
Man: http://www.dmtf.org./cim/mapping/http/v1.0 ; ns=23
23-CIMOperation: MethodCall
...
```

EXAMPLE 2:

Using POST:

```
POST /cimom HTTP/1.1
CIMOperation: MethodCall
...
```

6.3 Extension Headers Defined for CIM-XML Message Requests and Responses

A CIM-XML message contains exactly one CIM-XML operation request, CIM-XML operation response, CIM-XML export request, or CIM-XML export response. This clause describes the extension headers to specify CIM-XML message semantics in the HTTP header of a POST message.

Any CIM-XML operation request or CIM-XML operation response shall, and only CIM-XML operation requests and responses may, include the following CIM extension header:

- CIMOperation

Any CIM-XML operation request shall, and only CIM-XML operation requests may, include one and only one of the following CIM extension header sets:

- CIMMethod and CIMObject, or
- CIMBatch

Any CIM-XML export request or CIM-XML export response shall, and only CIM-XML export requests and responses may, include the following CIM extension header:

- CIMExport

Any CIM-XML export request shall, and only CIM-XML export requests may, include one and only one of the following CIM extension headers:

- CIMExportMethod
- CIMExportBatch

An HTTP response with an error status code to a CIM-XML message request may include the following CIM extension header:

- CIMError

All CIM-XML messages may include the following CIM extension header:

- CIMProtocolVersion

6.3.1 Encoding of CIM Element Names within HTTP Headers and Trailers

CIM element (class, property, qualifier, method, or method parameter) names are natively Unicode, and may use UCS-2 characters unsuitable for inclusion within an HTTP message header or trailer. To encode CIM element names represented in Unicode to values within HTTP headers or trailers, the following two-step mapping process shall be used:

- Encode the full Unicode CIM element name using UTF-8.
- Using the ""%" HEX HEX" convention, apply the standard URI [RFC2396, section 2] escaping mechanism to the resulting string to escape any characters that are unsafe within an HTTP header or trailer.

In this document, the token CIMIdentifier represents a CIM element name to which this transformation has been applied.

2878 One characteristic of this mapping is that CIM elements named with an ASCII representation appear in
2879 ASCII in the resulting URL.

2880 EXAMPLES:

2881 • CIM_LogicalElement is unchanged under this transformation.

2882 • The class named using the UCS-2 sequence representing the Hangul characters for the Korean
2883 word "hangugo" (D55C, AD6D, C5B4) becomes

2884 `%ED%95%9C%EA%B5%AD%EC%96%B4=10`

2885 after UTF-8 transformation and escaping all characters with their % HEX HEX equivalent.

2886 ### 6.3.2 Encoding of CIM Object Paths within HTTP Headers and Trailers

2887 This clause describes the mapping that shall be applied to represent CIM object paths, as described
2888 within an Operation Request Message using the <LOCALNAMESPACEPATH>, <LOCALCLASSPATH>,
2889 or <LOCALINSTANCEPATH> elements, in a format that is safe for representation within an HTTP header
2890 or trailer.

2891 If the element to be transformed is a <LOCALNAMESPACEPATH>, the algorithm is as follows:

2892 • For the first <NAMESPACE> child element, output the textual content of that element.

2893 • For each subsequent <NAMESPACE> child element, output the forward slash character (/)
2894 followed by the textual content of that <NAMESPACE> element.

2895 If the element to be transformed is a <LOCALCLASSPATH>, the algorithm is as follows:

2896 • Transform the <LOCALNAMESPACEPATH> child element using the rules previously
2897 described, and output a colon character (:).

2898 • Output the value of the NAME attribute of the <CLASSNAME> child element.

2899 If the element to be transformed is a <LOCALINSTANCEPATH>, the algorithm is as follows:

2900 • Transform the <LOCALNAMESPACEPATH> child element using the rules previously
2901 described, and output a colon character (:).

2902 • Output the value of the CLASSNAME attribute of the <INSTANCENAME> child element.

2903 • If there is at least one <KEYBINDING> child element under the <INSTANCENAME> child
2904 element, then for each such child element:

2905 – Output a period character (.) if this is the first <KEYBINDING> child element; otherwise,
2906 output a comma character (,).

2907 – Output the value of the NAME attribute, followed by an equal character (=).

2908 – If there is a <KEYVALUE> child element, output the textual element content of that
2909 element, subject to the following transformation:

2910 • If the VALUETYPE attribute is numeric or Boolean, the output is identical to the
2911 content of the element.

2912 • If the VALUETYPE attribute is a string, the output is obtained by enclosing the content
2913 of the element in double quote (") characters and escaping any double quote
2914 characters or backslash character within the value with a preceding backslash (\)
2915 character.

2916 – If there is a <VALUE.REFERENCE> child element

2917 • Output a double quote character (").

2918 • Apply the process recursively to the <CLASSPATH> or <INSTANCEPATH> child
2919 element of the <VALUE.REFERENCE> element, escaping any double quote or
2920 backslash character thereby generated with a preceding backslash (\) character.

2921 • Output a closing double quote character (").

2922 • If there is no <KEYBINDING> child element but there is a <KEYVALUE> or
2923 <VALUE.REFERENCE> child element under the <INSTANCENAME> child element, then:

2924 – Output an equal character (=).

2925 – Output the transformed value of the <KEYVALUE> or <VALUE.REFERENCE> using the
2926 previously-described rules.

2927 • If there are no <KEYBINDING> child elements or no <KEYVALUE> or <VALUE.REFERENCE>
2928 child element, then indicate a singleton instance by outputting the string "=@" under the
2929 <INSTANCENAME> child element.

2930 Finally, after applying these rules to the <LOCALNAMESPACEPATH>, <LOCALCLASSPATH>, or
2931 <LOCALINSTANCEPATH> element, transform the entire output string into URI-safe format in the
2932 following two-step procedure:

2933 • Encode the string using UTF-8 [RFC2279] if it is not already in this format.

2934 • Using the ""%" HEX HEX" convention, apply the standard URI [RFC2396, section 2] escaping
2935 mechanism to the resulting string to escape any characters that are unsafe within an HTTP
2936 header or trailer.

2937 In this document, the token CIMObjectPath represents a <LOCALNAMESPACEPATH>,
2938 <LOCALCLASSPATH>, or <LOCALINSTANCEPATH> element to which the preceding transformation
2939 has been applied.

2940 ### 6.3.3 CIMOperation

2941 The CIMOperation header shall be present in all CIM-XML operation request and CIM-XML operation
2942 response messages. It identifies the HTTP message as carrying a CIM-XML operation request or
2943 response.

2944 `CIMOperation = "CIMOperation" ":" ("MethodCall" | "MethodResponse")`

2945 A WBEM client shall include this header, with the value "MethodCall," in all CIM-XML operation requests
2946 that it issues. A WBEM server shall include this header in all CIM-XML operation responses that it issues,
2947 with the value "MethodResponse".

2948 If a WBEM server receives a CIM-XML operation request with this header, but with a missing value or a
2949 value that is not "MethodCall," then it shall fail the request with status "400 Bad Request". The WBEM
2950 server shall include a CIMError header in the response with a value of `unsupported-operation`.

2951 If a WBEM server receives a CIM-XML operation request without this header, it shall not process it as a
2952 CIM-XML operation request. The status code returned by the WBEM server in response to such a
2953 request is outside the scope of this document.

2954 If a WBEM client receives a response to a CIM-XML operation request without this header (or if this
2955 header has a value that is not "MethodResponse"), it should discard the response and take appropriate
2956 measures to publicize that it has received an incorrect response. The details as to how this is done are
2957 outside the scope of this document.

2958 The CIMOperation header affords a simple mechanism by which firewall or proxy administrators can
2959 make global administrative decisions on all CIM operations.

6.3.4 CIMExport

The CIMExport header shall be present in all CIM-XML export request and response messages. It identifies the HTTP message as carrying a CIM export method request or response.

```
CIMExport = "CIMExport" ":" ("MethodRequest" | "MethodResponse")
```

A WBEM client shall include this header with the value "MethodRequest" in all CIM-XML export requests that it issues. A WBEM listener shall include this header in all CIM-XML export responses that it issues, with the value "MethodResponse".

If a WBEM listener receives a CIM-XML export request with this header, but with a missing value or a value that is not "MethodRequest", then it shall fail the request with status "400 Bad Request". The WBEM listener shall include a CIMError header in the response with a value of unsupported-operation.

If a WBEM listener receives a CIM-XML export request without this header, it shall not process it. The status code returned by the WBEM listener in response to such a request is outside of the scope of this document.

If a WBEM client receives a response to a CIM-XML export request without this header (or if this header has a value that is not "MethodResponse"), it should discard the response and take appropriate measures to publicize that it has received an incorrect response. The details as to how this is done are outside the scope of this document.

The CIMExport header affords a simple mechanism by which firewall or proxy administrators can make global administrative decisions on all CIM exports.

6.3.5 CIMProtocolVersion

The CIMProtocolVersion header may be present in any CIM-XML message. The header identifies the version of the CIM operations over the HTTP specification in use by the sending entity.

```
CIMProtocolVersion = "CIMProtocolVersion" ":" 1*DIGIT "." 1*DIGIT
```

If the header is omitted, then a value of 1.0 must be assumed.

The major and minor revision numbers must be treated as independent integers.

The CIMProtocolVersion $x_1.y_1$ is less than CIMProtocolVersion $x_2.y_2$ if and only if one of the following statements is true:

- x_1 is less than x_2
- x_1 equals x_2, and y_1 is less than y_2

The CIMProtocolVersion $x_1.y_1$ is greater than CIMProtocolVersion $x_2.y_2$ if and only if one of the following statements is true:

- x_1 is greater than x_2,
- x_1 equals x_2, and y_1 is greater than y_2

A CIMProtocolVersion $x_1.y_1$ is within tolerance of CIMProtocolVersion $x_2.y_2$ if:

- x_1 equals x_2, and
- y_1 is less than or equal to y_2

If the CIMProtocolVersion of the CIM-XML message received is within tolerance of the CIMProtocolVersion supported for a WBEM server or WBEM listener implementation, the receiving implementation shall accept that CIM-XML message. Equivalent CIMProtocolVersion values between WBEM server or WBEM listener and the WBEM client shall be accepted. The WBEM server or WBEM

listener implementation may reject a CIM-XML message in all other cases. For information about how CIM-XML messages are rejected, see 7.3.

Beyond tolerance considerations, the implementation should reject the received CIM-XML message *only* if the design as defined by the CIMProtocolVersion of the receiving implementation has changed in the declaration of the API, method parameters, or behavior since the design defined by the CIMProtocolVersion of the received CIM-XML message.

6.3.6 CIMMethod

The CIMMethod header shall be present in any CIM-XML operation request message that contains a Simple Operation Request.

It shall not be present in any CIM-XML operation response message nor in any CIM-XML operation request message unless it is a simple operation request. It shall not be present in any CIM-XML export request or response message.

The header identifies the name of the CIM method to be invoked, encoded in an HTTP-safe representation. Firewalls and proxies may use this header to carry out routing and forwarding decisions based on the CIM method to be invoked.

The name of the CIM method within a simple operation request is the value of the `NAME` attribute of the <METHODCALL> or <IMETHODCALL> element.

```
CIMMethod  = "CIMMethod" ":" MethodName

MethodName = CIMIdentifier
```

If a WBEM server receives a CIM-XML operation request for which any one of the following statements is true, then it shall fail the request and return a status of "400 Bad Request". Also, it shall include a CIMError header in the response with a value of `header-mismatch`, subject to the considerations specified in 7.3:

- The CIMMethod header is present, but it has an invalid value.
- The CIMMethod header is not present, but the operation request message is a Simple Operation Request.
- The CIMMethod header is present, but the operation request message is not a simple operation request.
- The CIMMethod header is present and the operation request message is a simple operation request, but the CIMIdentifier value (when unencoded) does not match the unique method name within the simple operation request.

Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy was not acting on misleading information when it decided to forward the request based on the content of the CIMMethod header. Additional securing of HTTP messages against modification in transit (such as the encryption of the payload or appending of a digital signature thereto) would be required to provide a higher degree of integrity.

6.3.7 CIMObject

The CIMObject header shall be present in any CIM-XML operation request message that contains a Simple Operation Request.

It shall not be present in any CIM-XML operation response message nor in any CIM-XML operation request message unless it is a simple operation Request. It shall not be present in any CIM-XML export request or response message.

The header identifies the CIM object on which the method is to be invoked using a CIM object path encoded in an HTTP-safe representation. This object shall be a class or instance for an extrinsic method or a namespace for an intrinsic method. Firewalls and proxies may use this header to carry out routing and forwarding decisions based on the CIM object that is the target of a method invocation.

```
CIMObject  = "CIMObject" ":" ObjectPath

ObjectPath = CIMObjectPath
```

The ObjectPath value is constructed by applying the algorithm defined in 6.3.2 to either of the following child elements within the CIM-XML operation request:

- The <LOCALNAMESPACEPATH> child element of the <IMETHODCALL> element.
- The <LOCALCLASSPATH> or <LOCALINSTANCEPATH> child element of the <METHODCALL> element.

If a WBEM server receives a CIM-XML operation request for which any one of the following statements is true, then it shall fail the request and return a status of "400 Bad Request". Also, it shall include a CIMError header in the response with a value of `header-mismatch`, subject to the considerations specified in 7.3:

- The CIMObject header is present, but it has an invalid value.
- The CIMObject header is not present, but the operation request message is a Simple Operation Request.
- The CIMObject header is present, but the operation request message is not a simple operation request.
- The CIMObject header is present and the operation request message is a simple operation request, but the ObjectPath value does not match the operation request message (where a *match* is defined in 6.3.2).

Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy is not acting on misleading information when it forwards the request based on the content of the CIMObject header. Additional securing of HTTP messages against modification in transit, such as encrypting the payload or appending a digital signature to it, would be required to provide a higher degree of integrity.

6.3.8 CIMExportMethod

The CIMExportMethod header shall be present in any CIM-XML export request message that contains a simple export request.

This header shall not be present in any CIM-XML export response message nor in any CIM-XML export request message unless it is a simple export request. It shall not be present in any CIM-XML operation request or response message.

The CIMExportMethod header identifies the name of the CIM export method to be invoked, encoded in an HTTP-safe representation. Firewalls and proxies may use this header to carry out routing and forwarding decisions based on the CIM export method to be invoked.

The name of the CIM export method within a simple export request is the value of the NAME attribute of the <EXPMETHODCALL> element.

```
CIMExportMethod = "CIMExportMethod" ":" ExportMethodName

ExportMethodName = CIMIdentifier
```

If a WBEM listener receives a CIM-XML export request for which any one of the following statements is true, then it shall fail the request and return a status of "400 Bad Request". Also, it shall include a CIMError header in the response with a value of header-mismatch, subject to the considerations specified in 7.3:

- The CIMExportMethod header is present, but it has an invalid value.
- The CIMExportMethod header is not present, but the export request message is a simple export request.
- The CIMExportMethod header is present, but the export request message is not a simple export request.
- The CIMExportMethod header is present and the export request message is a simple export request, but the CIMIdentifier value (when unencoded) does not match the unique method name within the simple export request.

Note that this verification provides a basic level of assurance that any intermediate firewall or proxy is not acting on misleading information when it forwards the request based on the content of the CIMExportMethod header. Additional securing of HTTP messages against modification in transit, such as encrypting the payload or appending a digital signature to it, would be required to provide a higher degree of integrity.

6.3.9 CIMBatch

The CIMBatch header shall be present in any CIM-XML operation request message that contains a Multiple Operation Request.

This header shall not be present in any CIM-XML operation response message nor in any CIM-XML operation request message unless it is a multiple operation request. It shall not be present in any CIM-XML export request or response message.

The CIMBatch header identifies the encapsulated operation request message as containing multiple method invocations. Firewalls and proxies may use this header to carry out routing and forwarding decisions for batched CIM method invocations.

```
CIMBatch = "CIMBatch" ":"
```

If a WBEM server receives a CIM-XML operation request for which any one of the following statements is true, then it must fail the request and return a status of "400 Bad Request". Also it must include a CIMError header in the response with a value of `header-mismatch`, subject to the considerations specified in 7.3:

- The CIMBatch header is present, but it has an invalid value.
- The CIMBatch header is not present, but the operation request message is a multiple operation request.
- The CIMBatch header is present, but the operation request message is not a multiple operation request.

Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy is not acting on misleading information when it forwards the request based on the content of the CIMBatch header. Additional securing of HTTP messages against modification in transit, such as encrypting the payload or appending a digital signature to it, would be required to provide a higher degree of integrity.

If a WBEM server receives a CIM-XML operation request for which the CIMBatch header is present but the server does not support multiple operations, then it shall fail the request and return a status of "501 Not Implemented". Firewalls or Proxies may also employ this mechanism to compel a WBEM client to use simple operation requests rather than multiple operation requests.

A WBEM client that receives a response of "501 Not Implemented" to a multiple operation request should resubmit that request as a series of simple operation requests.

6.3.10 CIMExportBatch

The CIMExportBatch header shall be present in any CIM-XML export request message that contains a multiple export request.

It shall not be present in any CIM-XML operation request or response message. Also, it shall not be present in any CIM-XML export response message nor in any CIM-XML export request message unless it is a multiple export request.

The header identifies the encapsulated Export Request Message as containing multiple export method invocations. Firewalls and proxies may use this header to carry out routing and forwarding decisions for batched CIM Export method invocations.

```
CIMExportBatch = "CIMExportBatch" ":"
```

If a WBEM listener receives a CIM-XML export request for which any one of the following statements is true, then it must fail the request and return a status of "400 Bad Request". Also, it must include a CIMError header in the response with a value of header-mismatch, subject to the considerations specified in Errors:

- The CIMExportBatch header is present, but it has an invalid value.
- The CIMExportBatch header is not present, but the export request message is a multiple export request.
- The CIMExportBatch header is present, but the export request message is not a multiple export request.

Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy is not acting on misleading information when it forwards the request based on the content of the CIMExportBatch header. Additional securing of HTTP messages against modification in transit, such as encrypting the payload or appending a digital signature to it, would be required to provide a higher degree of integrity.

If a WBEM listener receives a CIM-XML export request for which the CIMExportBatch header is present, but the WBEM listener does not support multiple exports, then it shall fail the request and return a status of "501 Not Implemented". Firewalls or Proxies may also employ this mechanism to compel a WBEM client to use simple rather than multiple export requests.

A WBEM client that receives a response of "501 Not Implemented" to a multiple export request should resubmit that request as a series of simple export requests.

6.3.11 CIMError

The CIMError header may be present in any HTTP response to a CIM-XML message request that is not a CIM-XML message response.

It shall not be present in any CIM-XML message response or in any CIM-XML message request.

The CIMError header provides further CIM-specific diagnostic information if the WBEM server or WBEM listener encounters a fundamental error during processing of the CIM-XML operation request and is intended to assist clients to further disambiguate errors with the same HTTP status code:

```
CIMError  = "CIMError" ":" cim-error

cim-error = "unsupported-protocol-version" |
```

```
        "multiple-requests-unsupported" |
        "unsupported-cim-version" |
        "unsupported-dtd-version" |
        "request-not-valid" |
        "request-not-well-formed" |
        "request-not-loosely-valid" |
        "header-mismatch" |
        "unsupported-operation"
```

6.3.12 CIMRoleAuthenticate

A WBEM server may return a CIMRoleAuthenticate header as part of the 401 Unauthorized response along with the WWW-Authenticate header. The CIMRoleAuthenticate header must meet the challenge of indicating the WBEM server policy on role credentials.

```
challenge = "credentialrequired" | "credentialoptional" | "credentialnotrequired"
```

- A challenge of `credentialrequired` indicates that the WBEM server requires that a WBEM client must present a credential if it seeks to assume a role.
- A challenge of `credentialoptional` indicates that the credential is optional. If a credential is not sent, the WBEM server allows the role assumption if it is permitted for the given user. However, certain operations that require the role credential may not succeed.
- A challenge of `credentialnotrequired` indicates that no credential is required to assume the role.

Absence of the CIMRoleAuthenticate header indicates that the WBEM server does not support role assumption. A WBEM client should handle each of these cases appropriately.

The challenge does not contain any authorization scheme, realm, or other information. A WBEM client should extract this information from the WWW-Authenticate header. This implies that for any given request, the role credentials should use the same scheme as those required for the user credentials.

A WBEM server allows role assumption to succeed only if the user is allowed to assume the role. Therefore, even if appropriate credentials are presented, role assumption can fail. If either the user authentication or role assumption fails, the entire authentication operation fails.

To maintain backward compatibility, a WBEM server that supports role assumption must allow user authentication even if no role is specified.

6.3.13 CIMRoleAuthorization

The CIMRoleAuthorization header is supplied along with the normal authorization header that the WBEM client populates to perform user authentication. If the WBEM client needs to perform role assumption and the WBEM server challenge is credentialrequired, the CIMRoleAuthorization header must be supplied with the appropriate credentials. The credentials supplied as part of the CIMRoleAuthorization header must use the same scheme as those specified for the authorization header, as specified in RFC2617. Therefore, both Basic and Digest authentication are possible for the role credential.

If the WBEM client wishes to assume a role but does not wish to supply role credentials for server challenge credentialoptional or credentialnotrequired, the CIMRoleAuthorization header must set the auth-scheme field as specified in RFC2617 to be "role". The auth-param must contain the role name.

A WBEM server that supports roles must be capable of handling the presence of credentials in the CIMRoleAuthorization header (that is auth-scheme not set to "role") regardless of whether it is expecting credentials or not. It may choose to ignore these credentials.

6.3.14 CIMStatusCodeDescription

If a CIM product includes the CIMStatusCode trailer, it may also include the CIMStatusCodeDescription trailer. The value of this trailer is a string describing the nature of the error. A CIM product shall not include this trailer if the CIMStatusCode trailer is not present.

6.3.15 WBEMServerResponseTime

The WBEMServerResponseTime header may be present in any CIM response message. If it is present, the header shall contain a measure, specified in microseconds, of the elapsed time required by the WBEM server to process the request and create a response. Specifically, WBEMServerResponseTime describes the time elapsed since the WBEM server received the CIM request message and the associated CIM response message was ready to send to the WBEM client.

WBEMServerResponseTime = "WBEMServerResponseTime" ":", where the response time must be representable as a 64-bit unsigned integer value. If the actual elapsed time exceeds the maximum representable value, then the maximum value shall be returned. If the actual elapsed time is less than 1 microsecond, then a 0 shall be returned.

Although a WBEM client may ignore the WBEMServerResponseTime header, it shall allow this header to be included in a response.

7 HTTP Requirements and Usage

This clause describes HTTP support and the use of standard headers.

7.1 HTTP and HTTPS Support

CIM products shall support CIM-XML messages in HTTP. The following applies to this case:

- CIM products should support HTTP/1.1 as defined in RFC2616.

DEPRECATED

CIM products may support HTTP/1.0 as defined in RFC1945.

- Support for HTTP/1.0 is deprecated since version 1.4 of this document; HTTP/1.1 should be supported instead.

DEPRECATED

CIM products should support CIM-XML messages in HTTPS. If they do, the following applies to this case:

- CIM products shall support HTTPS as defined in RFC2818. This includes the use of HTTP within HTTPS, as defined in RFC2818.

  NOTE RFC2818 describes the use of TLS 1.0 and higher but not the use of SSL 2.0 or 3.0.

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

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

Requirements and considerations for authentication and encryption between CIM products are described in 7.4.

CIM products that use extension headers as defined in this document shall conform to the requirements defined in RFC2774 for their use.

7.2 Use of Standard HTTP Headers

Unless otherwise stated in this document, CIM products shall comply with the requirements on the use of standard HTTP headers described in RFC1945 and RFC2616. This clause defines only *additional* requirements on CIM products with respect to the use of these standard HTTP headers in a CIM-XML message.

Note that CIM products should not use HTTP headers defined in RFC2068 but deprecated in RFC2616 (for example, Public, Content-Base).

7.2.1 Accept

If a WBEM client includes an Accept header in a request, it shall specify a value that allows the WBEM server to return an entity body of "text/xml" or "application/xml" in the response.

A WBEM server or WBEM listener shall accept any value for this header stating that "text/xml" or "application/xml" is an acceptable type for a response entity. A WBEM server or WBEM listener should return "406 Not Acceptable" if the Accept header indicates that neither of these content types is acceptable.

If a WBEM server or WBEM listener accepts a request to return an entity of a type other than "text/xml" or "application/xml", the nature of the response is outside the scope of this document.

7.2.2 Accept-Charset

If a WBEM client includes an Accept-Charset header in a request, it shall specify a value that allows the WBEM server or WBEM listener to return an entity body using the character set "UTF-8".

A WBEM server or WBEM listener shall accept any value for this header asserting that "UTF-8" is an acceptable character set for a response entity. If the client does not provide an Accept-Charset, then "UTF-8" should be assumed by the WBEM server or WBEM listener.

```
Accept-Charset: UTF-8
```

A WBEM server or WBEM listener shall return "406 Not Acceptable" if the character set requested in the Accept-Charset header is not supported.

If a WBEM server or WBEM listener accepts a request to return an entity using a character set other than "UTF-8", the behavior of the subsequent WBEM client and WBEM server interaction is outside the scope of this document. See 7.8 for details.

7.2.3 Accept-Encoding

If a WBEM client includes an Accept-Encoding header in a request, it shall specify a q value that allows the WBEM server or WBEM listener to use the "Identity" encoding. The value shall be greater than 0 or not specified.

```
Accept-Encoding: Identity
Accept-Encoding: Identity; q=1.0
```

A WBEM server or WBEM listener shall accept any value for this header asserting that "Identity" is an acceptable encoding for the response entity.

A WBEM server or WBEM listener shall return "406 Not Acceptable" if the Accept-Encoding header indicates that the requested encoding is not acceptable.

7.2.4 Accept-Language

If a WBEM client includes an Accept-Language header in a request, it shall request a language-range, special-range, or both. The WBEM client shall also allow any language to be returned if the requested languages cannot be supported. This is accomplished by including the special-range, "*". The WBEM client may request multiple languages. Each language has equal priority, unless a q value is provided.

```
Accept-Language: zh, *
Accept-Language: zh;q=1.0, en;q=.7, *
```

Each CIM element in the response should be localized in only one language. A CIM element shall not be duplicated in the response because it is localized in more than one language.

WBEM servers may support multiple languages. A CIM product shall interpret the use of the special-range value, "*", as a request to return the response content using the default language defined for the target processing the request. Multiple targets, with different default language settings, may participate in the construction of a response. (See RFC2616 section 3.10 and ISO 639-1.)

See 7.8 for more information.

7.2.5 Accept-Ranges

WBEM clients shall not include the Accept-Ranges header in a request. A WBEM server or WBEM listener shall reject a request that includes an Accept-Range header with a status of "406 Not Acceptable".

7.2.6 Allow

If a WBEM server or WBEM listener is returning a "405 Method Not Allowed" response to a CIM-XML message request, then the Allow header shall include either M-POST or POST. Whether it includes any other HTTP methods is outside the scope of this document.

7.2.7 Authorization

See 7.4 for details.

7.2.8 Cache-Control

Generally, a CIM-XML message request may consist of a mixture of CIM method invocations, some of which may be eminently able to cache (for example, the manufacturer label on a disk drive) and some of which may be decidedly impossible to cache (for example, format a disk drive).

Furthermore, the encapsulation of such multiple method invocations in an HTTP POST or M-POST means that if a CIM-XML message request has any effect on an HTTP cache it is likely to be one of invalidating cached responses for the target WBEM server or WBEM listener. Indeed, HTTP/1.1 stipulates that by default POST responses cannot be cached unless the WBEM server indicates otherwise using an appropriate Cache-Control or Expires header.

For these reasons, CIM-XML message responses should not be considered as able to be cached. A WBEM server or WBEM listener should not include a Cache-Control header in a CIM-XML message response that might indicate to a cache that the response can be cached.

If the WBEM server or WBEM listener is responding to a CIM-XML message request conveyed in an M-POST request, then in accordance with RFC2774 the WBEM server or WBEM listener shall include a no-cache control directive to prevent inadvertent caching of the "Ext" header, as in the following example:

EXAMPLE

```
HTTP/1.1 200 OK
Ext:
Cache-Control: no-cache
...
```

7.2.9 Connection

The following courses of action are recommended for connections:

- WBEM clients should avoid the use of the "Connection: close" header unless it is known in advance that this is the only request likely to be sent out on that connection.
- WBEM servers and WBEM listener support persistent connections wherever possible.

Timeout mechanisms should be employed to remove idle connections on the WBEM client, WBEM server, and WBEM listener. The details of timeout mechanisms are outside the scope of this document. Clients should be cautious in retrying requests, especially if they are not idempotent (for example, method invocation).

WBEM clients, WBEM servers, and WBEM listeners should support pipelining (HTTP/1.1 only, see RFC2616) if possible, but be aware of the requirements defined in RFC2616. In particular, attention is drawn to the requirement from RFC2616 that clients not pipeline requests using non-idempotent methods or non-idempotent sequences of methods. A client that needs to send a non-idempotent request should wait to send that request until it receives the response status for the previous request.

7.2.10 Content-Encoding

If a WBEM client includes a Content-Encoding header in a request, it should specify a value of "identity", unless there is good reason to believe that the WBEM server or WBEM listener can accept another encoding.

7.2.11 Content-Language

The Content-Language entity-header field of a CIM-XML message describes the natural language(s) of the intended audience of the content.

A CIM-XML message may contain a Content-Language header. The value of the Content-Language header in a CIM response message shall be consistent with the Accept-Language values specified in the corresponding CIM request message. If the WBEM server cannot determine one or more of the content languages used to construct the response, then the Content-Language entity shall not be returned.

Multiple targets using different Content-Language values may participate in constructing a response. The Content-Language field shall reflect all Content-Language values used to construct the response. The content of a CIM-XML message may contain elements in languages not listed in the Content-Language field.

```
Content-Language: en
```

See 7.8 for details.

7.2.12 Content-Range

WBEM clients, WBEM servers, and WBEM listeners shall not use this header.

7.2.13 Content-Type

WBEM clients, WBEM servers, and WBEM listeners shall specify (and accept) a media type for the Content-Type header of either "text/xml" or "application/xml" as defined in RFC2376. In addition, they may specify and shall accept a "charset" parameter as defined in RFC2616. If a "charset" parameter is specified, it shall have the value "utf-8" either with or without surrounding double quotes. The sending side should use the form without double quotes. The receiving side shall support both forms. If a "charset" parameter is not specified, the receiving side shall assume "utf-8" as a default.

Examples of valid Content-Type headers are:

```
Content-type: text/xml
Content-type: text/xml; charset=utf-8
Content-type: text/xml; charset="utf-8"
Content-type: application/xml
Content-type: application/xml; charset=utf-8
Content-type: application/xml; charset="utf-8"
```

7.2.14 Expires

For the reasons described in 7.2.8, a WBEM server or WBEM listener shall not include an Expires header in a CIM-XML message response that might indicate to a cache that the response can be cached.

7.2.15 If-Range

WBEM clients, WBEM servers, and WBEM listeners shall not use this header.

7.2.16 Proxy-Authenticate

See 7.4 for details.

7.2.17 Range

WBEM clients, WBEM servers, and WBEM listeners shall not use this header.

7.2.18 WWW-Authenticate

See 7.4 for details.

7.3 Errors and Status Codes

This clause defines how WBEM servers and WBEM listeners shall handle errors that occur in processing a CIM-XML message request. This document does not introduce any new HTTP response status codes.

If there is an error in processing the HTTP Request-Line or standard HTTP headers, the WBEM server or WBEM listener shall take appropriate action as dictated by its conformance to the relevant version of HTTP (see 7.1).

Otherwise, if there are any mandatory extension declarations that the WBEM server does not support it shall respond with a "510 Not Extended" status according to RFC2774.

Otherwise, the request shall be processed in accordance with the relevant version of HTTP (see 7.1) and the additional rules defined in this document.

Assuming that the HTTP request is otherwise correct, the WBEM server or WBEM listener shall use the following status codes when processing the CIM extension headers:

- 501 Not Implemented

This status code indicates that one of the following situations occurred:

- The CIMProtocolVersion extension header in the request specifies a version of the CIM mapping onto HTTP that is not supported by this WBEM server or WBEM listener. The WBEM server or WBEM listener shall include a CIMError header in the response with a value of `unsupported-protocol-version`.
- The client specified a Multiple Operation Request (or multiple Export Request), and the WBEM server (or WBEM listener) does not support such requests. The WBEM server or WBEM listener shall include a CIMError header in the response with a value of `multiple-requests-unsupported`.
- The CIMVERSION attribute in the message request is not set to a proper value. The CIMVERSION attribute shall be in the form of "M.N", where M is the major revision of the specification in numeric form and N is the minor revision in numeric form. The version shall be at "2.0" or greater (for example, "2.0" or "2.3"). The WBEM server or WBEM listener shall include a CIMError header in the response with a value of `unsupported-cim-version`.
- The DTDVERSION attribute in the message request is not set to a proper value. The DTDVERSION attribute shall be in the form of "M.N", where M is the major revision of the specification in numeric form and N is the minor revision in numeric form. The version shall be at "2.0" or greater (for example, "2.0" or "2.1"). The WBEM server or WBEM listener shall include a CIMError header in the response with a value of `unsupported-dtd-version`.

- 401 Unauthorized

  The WBEM server or WBEM listener is configured to require that a client authenticate itself before it can issue CIM-XML message requests to the WBEM server or WBEM listener.

- 403 Forbidden

  The WBEM server or WBEM listener does not allow the client to issue CIM-XML message requests. The WBEM server or WBEM listener may alternatively respond with a "404 Not Found" if it does not wish to reveal this information to the client.

- 407 Proxy Authentication Required

  The WBEM server or WBEM listener is configured to require that the proxy authenticate itself before it can issue CIM-XML message requests on behalf of a WBEM client to the WBEM server or WBEM listener.

Assuming that the CIM extension headers are correct, a validating WBEM server or WBEM listener (one that enforces the validity of the CIM-XML message request with respect to the CIM XML DTD) shall use the following status code when processing the entity body containing the CIM-XML message request:

- 400 Bad Request

  The entity body defining the CIM-XML message request is not well-formed or not valid with respect to the CIM XML DTD. The WBEM server or WBEM listener shall include a CIMError header in the response with a value of `request-not-well-formed` or `request-not-valid` (as appropriate).

A loosely-validating WBEM server or WBEM listener only enforces the CIM-XML message request to be loosely valid. Therefore, it may reject a CIM-XML message request that is not loosely valid with an HTTP status code of 400 (Bad Request) before further processing. In this case, the WBEM server or WBEM listener shall include a CIMError header in the response with a value of `request-not-loosely-valid`.

A loosely-validating WBEM server or WBEM listener shall reject a CIM-XML message request that is not well-formed with an HTTP status code of 400 (Bad Request). In this case, the WBEM server or WBEM listener shall include a CIMError header in the response with a value of `request-not-well-formed`.

A loosely-validating WBEM server or WBEM listener shall not reject an invalid CIM-XML message request that is loosely valid in the XML sense.

A loosely-validating WBEM server or WBEM listener shall ultimately signal an error to the WBEM client if the CIM-XML message request is not loosely valid. That is, the request is missing required content or the required content is incorrect, such as an attribute with an invalid value according to the CIM XML DTD. It is not mandated to reject a CIM-XML message request before processing, for to do otherwise would compel the WBEM server or WBEM listener to check the complete request before processing can begin and this would be as expensive as requiring the WBEM server or WBEM listener to fully validate the request. Therefore, a loosely-validating server or listener may elect to begin processing the request and issuing a response (with an HTTP success status code) before verifying that the entire request is loosely valid.

A WBEM client may use the CIMValidation header mechanism to determine whether a WBEM server or WBEM listener is validating or loosely-validating.

Assuming that the CIM-XML message request is correctly formed as previously described, the WBEM server or WBEM listener shall process the request accordingly and return a CIM-XML message response.

The entity body shall be a correct CIM-XML message response for that request.

If the CIM-XML message response contains an entity that is a simple message response, then the response status shall be "200 OK". Otherwise, the response status shall be "207 Multistatus".

7.4 Security Considerations

This subclause describes requirements and considerations for authentication and message encryption between CIM products.

7.4.1 Authentication

This subclause describes requirements and considerations for authentication between CIM products. Specifically, authentication happens from WBEM clients to WBEM servers for CIM-XML operation messages, and from WBEM servers to WBEM listeners for CIM-XML export messages. The authentication mechanisms defined in this subclause apply to both HTTP and HTTPS.

CIM products may support operating without the use of authentication. This practice is not recommended and should only be done in environments where lack of network privacy is not an issue (for example, in a physically secure private network or on the same operating system).

Basic authentication is described in RFC1945 and RFC2068. Digest authentication is defined in RFC2069. Both authentication schemes are covered in a consolidated document (RFC2617), which also makes a number of improvements to the original specification of digest authentication. This document requires conformance to RFC2617 but not to the earlier documents.

Basic authentication provides a very rudimentary level of authentication, with the major weakness that the client password is sent over the wire in unencrypted form (unless HTTPS is used)..

CIM products may support basic authentication as defined in RFC2617. Basic authentication without HTTPS should only be used in environments where lack of network privacy is not an issue.

Digest authentication verifies that both parties share a common secret without having to send that secret.

CIM products should support digest authentication as defined in RFC2617.

CIM products may support authentication mechanisms not covered by RFC2617. One example are public key certificates as defined in X.509.

WBEM servers and WBEM listeners should require that WBEM clients and WBEM servers, respectively, authenticate themselves. This document does not mandate this because it is recognized that in some circumstances the WBEM server or WBEM listener may not require or wish the overhead of employing authentication. WBEM servers and WBEM listeners should carefully consider the performance/security tradeoffs in determining how often to issue challenges to WBEM clients and WBEM servers, respectively.

A WBEM server or WBEM listener that returns a "401 Unauthorized" response to a CIM message request shall include one WWW-Authenticate response-header indicating one supported authentication mechanism. This document does not mandate use of basic or digest authentication because it is recognized that in some circumstances the WBEM server or WBEM listener may use bespoke authentication mechanisms not covered by RFC2617. Similar considerations apply to the use of the Proxy-Authenticate response-header in "407 Proxy Authentication Required".

7.4.2 Message Encryption

Encryption of messages between CIM products is supported by the use of HTTPS in the communication between CIM products. Requirements for the use of HTTPS and its underlying secure sockets are defined in 7.1.

The following requirements on cipher suites apply to CIM products that support 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 800-57 and NIST 800-131A)
- Any additional cipher suites may be supported

7.5 Determining WBEM server Capabilities

If a WBEM server can return capabilities information, there are two techniques for returning this information as defined in this document:

- The preferred technique is through the use of the classes defined in 7.5.1.
- Alternatively, use of the HTTP OPTIONS method as defined in 7.5.2 is allowed because historically it is the original technique defined for requesting capabilities information.

Use of the CIM classes defined in 7.5.1 is strongly encouraged and it is expected that this method will be enhanced and extended in the future to provide more capabilities information. The future use of the HTTP OPTIONS method to determine capabilities of WBEM servers is discouraged. It will probably not be expanded significantly and may be reviewed for possible deprecation in the next major revision of this document.

7.5.1 Determining WBEM server Capabilities through CIM Classes

A set of CIM classes is defined specifically to return WBEM server capabilities information as follows:

- CIM_ObjectManager

  This class is a type of CIM_Service that defines the capabilities of the target WBEM server.

- CIM_ObjectManagerCommunicationMechanism

  This class describes access to the target WBEM server. It defines the capabilities of the WBEM server that are available through the target Object Manager Communication mechanism. A WBEM server is allowed to support different capabilities through different communication mechanisms.

- CIM_CIMXMLCommunicationMechanism

  This class specializes on ObjectManagerCommunicationMechanism, adding properties specific to the CIM-XML encoding and protocol.

- CIM_CommMechanismForManager

  This association between CIM_ObjectManager and CIM_ObjectManagerCommunicationMechanism defines the communications protocols (and corresponding capabilities) available on the target WBEM server through the ObjectManagerCommunicationMechanism instances.

A WBEM client may use instances of these CIM classes to determine the CIM capabilities (if any) of the target WBEM server. A WBEM server that supports capabilities determination through these classes shall support at least the Enumerate Instance and Get Instance operations for the classes. The use of other methods of the basic read profile is optional. A WBEM server that does not support the determination of CIM capabilities through these classes shall return CIM_ERR_NOT_FOUND to any instance or class request on these classes. These classes shall not be used for reporting any other information than capabilities of the target WBEM server.

To provide interoperability, the CIM object manager classes shall exist in a well-known namespace. Because there is no discovery mechanism that can define this well-known namespace to a WBEM client, it shall be one or more predefined namespaces. Therefore, to ensure interoperability, we recommend that pending future extensions of the WBEM specifications include discovery tools that define a namespace for these classes in a WBEM server; these predefined namespaces should exist in either the root namespace or in the /root/CIMV2 namespace.

A WBEM server that supports capabilities reporting through these classes shall correctly report the current actual capabilities of the target WBEM server and shall report on all capabilities defined. A WBEM server is allowed to report "none" if the capability does not exist or "unknown" if the status of the capability is unknown at the time of the request for those properties where these choices exist in the properties definition. Because the CIM_ObjectManager object provides information on the target WBEM server, only a single instance of this class may exist in a WBEM server.

The capabilities to be reported through the CIM_ObjectManagerCommunicationMechanism are as follows:

- CommunicationMechanism property, which defines the communication protocol for the CommunicationMechanism object. A compliant WBEM server shall include the CIM-XML protocol for at least one ObjectManagerCommunicationMechanism instance.

- ProfilesSupported property, which defines the functional profiles supported as defined in clause 5.4.4. All WBEM servers shall support the basic-read functional group. All WBEM clients may assume that any WBEM server supports the basic-read functional group. The list of functional groups returned by a WBEM server shall contain the basic-read group and shall not contain

duplicates. WBEM clients shall ignore duplicate entries in the functional-group list. If a functional group is included in the list, the WBEM client shall assume that all other groups on which it depends (according to the rules defined in 5.4.4) are also supported. A WBEM server should not explicitly include a functional group in the list whose presence may be inferred implicitly by a dependency. Support for a functional group does not imply that any method from that group will always succeed. Rather, the absence of the functional group from this list (whether explicit or implied) indicates to the WBEM client that methods in that group will never succeed.

- MultipleOperationsSupported property, which defines whether the target WBEM server supports multiple operation requests as defined in 5.4.2. `True` in this property indicates that the WBEM server can accept and process multiple operation requests. `False` indicates that the WBEM server can accept only single operation requests.
- AuthenticationMechanismsSupported property, which defines the authentication mechanisms supported by the target WBEM server as defined in 7.4.
- PulledEnumerationClosureOnExceedingServerLimits property, which indicates whether the WBEM server supports closure of Pulled Enumeration sessions based upon exceeding server limits.
- PulledEnumerationContinuationOnErrorSupported property, which indicates whether the WBEM server supports continuation on error for Pulled enumerations.
- PulledEnumerationMinimumOperationTimeout (PulledEnumerationMaximumOperationTimeout) property, which indicates the minimum (maximum) operation timeout allowed by the WBEM server for Pulled enumerations.

Compliant WBEM servers may report additional capabilities for the CommunicationMechanism Functional Profiles, QueryLanguageSupported, and AuthenticationMechanismSupported by defining the "other" enumeration in the property and returning additional information in the associated "additional capabilities" property.

7.5.2 Determining WBEM server Capabilities through the HTTP Options

A WBEM client may use the OPTIONS method to determine the CIM capabilities (if any) of the target server. A WBEM server may support the OPTIONS method (for example, WBEM servers supporting only HTTP/1.0 would not support OPTIONS).

To support the ability for a WBEM server to declare its CIM capabilities independently of HTTP, the DMTF intends to publish a CIM schema (in a separate document) describing such capabilities. In particular, this mechanism would allow servers that do not support the OPTIONS method to declare their capabilities to a client.

If a WBEM server supports the OPTIONS method, it should return the following headers in the response:

- CIM Extension Header CIMProtocolVersion, which provides a way for a client to discover the version of the CIM HTTP mapping supported by the WBEM server.
- CIM Extension Header CIMSupportedFunctionalGroups, which provides a way for a client to discover the CIM operations supported by the WBEM server.
- CIM Extension Header CIMSupportsMultipleOperations, which provides a way for the client to discover whether the WBEM server can support Multiple Operation Requests.

In addition, if the WBEM server supports one or more query languages for the ExecQuery operation (see 5.4.2.13), it should return the following header in the response:

- CIM Extension Header CIMSupportedQueryLanguages, which allows the client to discover the query languages supported by the WBEM server for the ExecQuery operation.

In addition, if the WBEM server runs in a fixed validation mode, it should return the following header in the response:

- CIM Extension Header CIMValidation, which allows the client to determine whether the WBEM server is strictly validating or loosely validating.

If the CIMProtocolVersion, CIMSupportedFunctionalGroups, CIMSupportsMultipleOperations, CIMValidation, or CIMSupportedQueryLanguages extension headers are included in the response, the WBEM server shall declare them as optional extension headers using the "Opt" header defined in RFC2774.

The full format of the "Opt" header declaration for this document is:

```
Opt              = "Opt" ":" "http://www.dmtf.org/cim/mapping/http/v1.0"
                   ";" "ns" "=" header-prefix

header-prefix    = 2*DIGIT
```

This header-prefix should be generated at random on a per-HTTP message basis and should not necessarily be a specific number.

EXAMPLE: The following is a fragment of a legitimate OPTIONS response from a WBEM server:

```
HTTP/1.1 200 OK
Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=77
77-CIMProtocolVersion: 1.0
77-CIMSupportedFunctionalGroups: basic-read
77-CIMBatch
77-CIMSupportedQueryLanguages: wql
...
```

7.5.2.1 CIMSupportedFunctionalGroups

The CIMSupportedFunctionalGroups extension header should be returned by a WBEM server in any OPTIONS response. It shall not be returned in any other scenario.

This header is defined as follows:

```
CIMSupportedFunctionalGroups = "CIMSupportedFunctionalGroups" ":"
                               1#functional-group

functional-group = "basic-read" |
   "basic-write" |
   "schema-manipulation" |
   "instance-manipulation" |
   "qualifier-declaration" |
   "association-traversal" |
   "query-execution"
```

The functional group definitions correspond directly to those listed in 5.5.3. All WBEM servers shall support the basic-read functional group. All WBEM clients may assume that any WBEM server supports the basic-read functional group.

The list of functional groups returned by a WBEM server shall contain the basic-read group and shall not contain any duplicates. WBEM clients shall ignore any duplicate entries in the functional-group list.

If a functional group is included in the list, the WBEM client shall assume that all other groups on which it depends (according to the rules defined in 5.5.3) are also supported. A WBEM server should not explicitly include a functional group in the list if the presence of the group may be implied by a dependency.

EXAMPLE: The following HTTP response message indicates that the WBEM server supports instance-manipulation, association-traversal, basic-write, and basic-read.

```
HTTP/1.1 200 OK
Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=77
77-CIMProtocolVersion: 1.0
77-CIMSupportedFunctionalGroups: association-traversal, instance-manipulation
...
```

Support for a functional group does *not* imply that any method from that group will always succeed. Rather, the absence (whether explicit or implied) of the functional group from this header is an indication to the WBEM client that methods in that group will *never* succeed.

7.5.2.2 CIMSupportsMultipleOperations

The CIMSupportsMultipleOperations extension header shall be returned in an OPTIONS response by any WBEM server that supports Multiple Operation Requests. It shall not be returned in any other circumstances.

This header is defined as follows:

```
CIMSupportsMultipleOperations  = "CIMSupportsMultipleOperations"
```

The presence of this header indicates that the WBEM server can accept and process multiple operation requests. The absence of this header indicates that the WBEM server can only accept and process Simple Operation Requests.

7.5.2.3 CIMSupportedQueryLanguages (DEPRECATED)

The CIMSupportedQueryLanguages extension header identifies the query languages supported by the WBEM server for the ExecQuery operation (see 5.4.2.13).

DEPRECATION NOTE: The CIMSupportedQueryLanguages extension header has been deprecated in version 1.4 of this document, because it was used only for the ExecQuery operation.

The CIMSupportedQueryLanguages extension header should be returned in any OPTIONS response by a WBEM server that supports at least one such query language. It shall not be returned in any other scenario.

This header is defined as follows (`token` has the meaning conferred by RFC1945 and RFC2616):

```
CIMSupportedQueryLanguages  = "CIMSupportedQueryLanguages" ":" 1#query-language

query-language     = token
```

The `query-language` value shall be treated as case-insensitive. It is anticipated that query languages will be submitted for approval to the DMTF, and each submission will define a value for this token to enable it to be specified in this header.

7.5.2.4 CIMValidation

The CIMValidation extension header may be returned by a WBEM server to provide information about the level of validation of CIM-XML operation request messages.

This header is defined as follows:

```
CIMValidation = "CIMValidation" ":" validation-level

validation-level = "validating" | "loosely-validating"
```

A validation-level of `validating` indicates that the WBEM server always applies strict validation of each CIM-XML operation request. A validation-level of `loosely-validating` indicates that the WBEM server applies loose validation of each CIM-XML operation request.

In the absence of this header, a WBEM client should assume that the WBEM server operates in strict validation mode.

7.6 Other HTTP Methods

This document does not in any way define or constrain the way a WBEM client, WBEM server, or WBEM listener uses any HTTP method other than those explicitly cited.

7.7 Discovery and Addressing

The target URI of the CIM-XML operation request is defined as the location of the WBEM server. This document does not constrain the format of this URI other than it should be a valid URI (RFC2396) for describing an HTTP-addressable resource.

An HTTP server that supports the CIM mapping defined in this document, and which supports the OPTIONS method, should include the following CIM extension header in an OPTIONS response:

- CIMOM

This header is defined as follows:

```
CIMOM        = "CIMOM" ":" (absoluteURI | relativeURI)
```

The terms `absoluteURI` and `relativeURI` are taken from RFC2616; they indicate the location of the WBEM server for this HTTP server.

If the CIMOM extension header is included in the response, the WBEM server shall declare it an optional extension header as described in 7.5.

A WBEM client that needs to communicate with a WBEM server on an HTTP server should try an OPTIONS request to that HTTP server. If the OPTIONS request fails or the response does not include the CIM-CIMOM extension header, the WBEM client may assume that the value of CIM-CIMOM is the relative URI `cimom`.

The DMTF recommends the use of the following well-known IP ports in compliant WBEM servers. This is a recommendation and not a requirement. The DMTF has registered these port addresses with IANA, so they are for the exclusive use of the DMTF.

- CIM-XML (HTTP) 5988/tcp
- CIM-XML (HTTP) 5988/udp
- CIM-XML (HTTPS) 5989/tcp
- CIM-XML (HTTPS) 5989/udp

Other discovery mechanisms are outside the scope of this version of the specification.

EXAMPLE 1:

This example shows an HTTP server located at http://www.dmtf.org/ issuing an OPTIONS response to an HTTP client to indicate that its WBEM server is located at http://www.dmtf.org/access/cimom.

```
HTTP/1.1 200 OK
Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=48
48-CIMOM: /access/cimom
...
```

EXAMPLE 2:

If an HTTP server located at http://www.dmtf.org/ responds with a "501 Not Implemented" to an OPTIONS request from a WBEM client, the WBEM client may then try to contact the WBEM server at http://www.dmtf.org/cimom.

7.8 Internationalization Considerations

This clause defines the capabilities of the CIM HTTP mapping with respect to IETF policy guidelines on character sets and languages (RFC2277).

In this document, human-readable fields are contained within a response or request entity body. In all cases, a human-readable content is encoded using XML (which explicitly provides for character set tagging and encoding) and requires that XML processors read XML elements encoded, at minimum, using the UTF-8 (RFC2279) encoding of the ISO 10646 multilingual plane.

Properties that are not of type string or string array shall not be localized.

Because keys are writeable only on instantiation, key values shall not be localized. See DSP0004 for details.

XML examples in this document demonstrate the use of the charset parameter of the Content-Type header, as defined in RFC2616, as well as the XML attribute on the <?xml> processing instruction, which together provide charset identification information for MIME and XML processors. This document mandates that conforming applications shall support at least the "UTF-8" charset encoding (RFC2277) in the Content-Type header and shall support the "UTF-8" value for the XML `encoding` attribute.

XML also provides a language tagging capability for specifying the language of the contents of a particular XML element, based on use of IANA registered language tags (RFC1766) in combination with ISO 639-1, in the xml:lang attribute of an XML element to identify the language of its content and attributes. Section 3.10 of RFC2616 defines how the two-character ISO 639-1 language code is used as the primary-tag. The language-tag shall be registered by IANA.

DSP0201 declares this attribute on any XML elements. Therefore, conforming applications should use this attribute when specifying the language in which a particular element is encoded for string and string array attributes and qualifiers. See the usage rules on this element, which are defined by the World Wide Web Consortium in *XML 1.0, second edition*. The attribute may be scoped by the instance or a class and should not be scoped by a property because instances or classes should be localized in one language.

This document defines several names of HTTP headers and their values. These names are constructed using standard encoding practices so that they always have an HTTP-safe ASCII representation. Because these headers are not usually visible to users, they do not need to support encoding in multiple character sets.

DSP0201 introduces several XML element names. Similarly, these names are not visible to an end user and do not need to support multiple character set encodings.

The CIM model (DSP0004) defines the subset of the Unicode character set that can be used to name CIM elements (classes, instances, methods, properties, qualifiers, and method parameters). In general, these characters appear as the value of XML attributes or as element content and are not displayed to end users.

Negotiation and notification of language settings is effected in this mapping using the standard Accept-Language and Content-Language headers defined in RFC1945 and RFC2616.

ANNEX A
(Informative)

Examples of Message Exchanges

This annex illustrates the protocol defined in this document with examples of valid HTTP request/response exchanges. The examples are for illustration purposes only and are not considered part of the specification.

For clarity, additional white space is included in the examples, but such white space is not an intrinsic part of such XML documents.

A.1 Retrieval of a Single Class Definition

The following HTTP request illustrates how a client requests the class CIM_VideoBIOSElement.

```
M-POST /cimom HTTP/1.1
HOST: http://www.myhost.com/
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: GetClass
73-CIMObject: root/cimv2

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLEREQ>
         <IMETHODCALL NAME="GetClass">
            <LOCALNAMESPACEPATH>
               <NAMESPACE NAME="root"/>
               <NAMESPACE NAME="cimv2"/>
            </LOCALNAMESPACEPATH>
            <IPARAMVALUE NAME="ClassName">
               <CLASSNAME NAME="CIM_VideoBIOSElement"/>
            </IPARAMVALUE>
            <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
         </IMETHODCALL>
      </SIMPLEREQ>
   </MESSAGE>
</CIM>
```

Following is an HTTP response to the preceding request indicating success of the requested operation. For clarity of exposition, the complete definition of the returned <CLASS> element is not shown.

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
```

```
73-CIMOperation: MethodResponse

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLERSP>
         <IMETHODRESPONSE NAME="GetClass">
            <IRETURNVALUE>
               <CLASS NAME="CIM_VideoBIOSElement"
                      SUPERCLASS="CIM_SoftwareElement">
                  ...
               </CLASS>
            </IRETURNVALUE>
         </IMETHODRESPONSE>
      </SIMPLERSP>
   </MESSAGE>
</CIM>
```

A.2 Retrieval of a Single Instance Definition

The following HTTP request illustrates how a client requests the instance MyClass.MyKey="S3".

```
M-POST /cimom HTTP/1.1
HOST: http://www.myhost.com/
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: GetInstance
73-CIMObject: root%2FmyNamespace

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="1.1">
   <MESSAGE ID="87855" PROTOCOLVERSION="1.0">
      <SIMPLEREQ>
         <IMETHODCALL NAME="GetInstance">
            <LOCALNAMESPACEPATH>
               <NAMESPACE NAME="root"/>
               <NAMESPACE NAME="myNamespace"/>
            </LOCALNAMESPACEPATH>
            <IPARAMVALUE NAME="InstanceName">
               <INSTANCENAME CLASSNAME="MyClass">
                  <KEYBINDING NAME="MyKey"><KEYVALUE>S3</KEYVALUE></KEYBINDING>
               </INSTANCENAME>
            </IPARAMVALUE>
            <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
         </IMETHODCALL>
      </SIMPLEREQ>
   </MESSAGE>
</CIM>
```

Following is an HTTP response to the preceding request indicating an error because the specified instance is not found.

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodResponse

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87885" PROTOCOLVERSION="1.0">
      <SIMPLERSP>
         <IMETHODRESPONSE NAME="GetInstance">
            <ERROR CODE="6" DESCRIPTION="Instance of MyClass not found"/>
         </IMETHODRESPONSE>
      </SIMPLERSP>
   </MESSAGE>
</CIM>
```

A.3 Deletion of a Single Class Definition

The following HTTP request illustrates how a client deletes the class CIM_VideoBIOSElement.

```
M-POST /cimom HTTP/1.1
HOST: http://www.myhost.com/
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: DeleteClass
73-CIMObject: root/cimv2

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLEREQ>
         <IMETHODCALL NAME="DeleteClass">
            <LOCALNAMESPACEPATH>
               <NAMESPACE NAME="root"/>
               <NAMESPACE NAME="cimv2"/>
            </LOCALNAMESPACEPATH>
            <IPARAMVALUE NAME="ClassName">
               <CLASSNAME NAME="CIM_VideoBIOSElement"/>
            </IPARAMVALUE>
         </IMETHODCALL>
      </SIMPLEREQ>
   </MESSAGE>
</CIM>
```

Following is an HTTP response to the preceding request indicating failure of the preceding operation due to the inability to delete instances of the class.

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
```

```
Content-Length: xxxx
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodResponse

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLERSP>
         <IMETHODRESPONSE NAME="DeleteClass">
            <ERROR CODE="9" DESCRIPTION="Class has non-deletable instances"/>
         </IMETHODRESPONSE>
      </SIMPLERSP>
   </MESSAGE>
</CIM>
```

A.4 Deletion of a Single Instance Definition

The following HTTP request illustrates how a client deletes the instance MyClass.MyKey="S3".

```
M-POST /cimom HTTP/1.1
HOST: http://www.myhost.com/
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: DeleteInstance
73-CIMObject: root%2FmyNamespace

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLEREQ>
         <IMETHODCALL NAME="DeleteInstance">
            <LOCALNAMESPACEPATH>
               <NAMESPACE NAME="root"/>
               <NAMESPACE NAME="myNamespace"/>
            </LOCALNAMESPACEPATH>
            <IPARAMVALUE NAME="InstanceName">
               <INSTANCENAME CLASSNAME="MyClass">
                  <KEYBINDING NAME="MyKey">
                     <KEYVALUE>S3</KEYVALUE>
                  </KEYBINDING>
               </INSTANCENAME>
            </IPARAMVALUE>
         </IMETHODCALL>
      </SIMPLEREQ>
   </MESSAGE>
</CIM>
```

Following is an HTTP response to the preceding request indicating success of the preceding operation.

```
HTTP/1.1 200 OK
```

```
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/operation ; ns=73
73-CIMOperation: MethodResponse

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLERSP>
         <IMETHODRESPONSE NAME="DeleteInstance"/>
      </SIMPLERSP>
   </MESSAGE>
</CIM>
```

A.5 Creation of a Single Class Definition

The following HTTP request illustrates how a client creates the class MySchema_VideoBIOSElement as a subclass of CIM_VideoBIOSElement. For clarity of exposition, most of the submitted <CLASS> element is omitted from the example.

```
M-POST /cimom HTTP/1.1
HOST: http://www.myhost.com/
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: CreateClass
73-CIMObject: root/cimv2

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLEREQ>
         <IMETHODCALL NAME="CreateClass">
            <LOCALNAMESPACEPATH>
               <NAMESPACE NAME="root"/>
               <NAMESPACE NAME="cimv2"/>
            </LOCALNAMESPACEPATH>
            <IPARAMVALUE NAME="NewClass">
               <CLASS NAME="MySchema_VideoBIOSElement"
                      SUPERCLASS="CIM_VideoBIOSElement">
                  ...
               </CLASS>
            </IPARAMVALUE>
         </IMETHODCALL>
      </SIMPLEREQ>
   </MESSAGE>
</CIM>
```

Following is an HTTP response to the preceding request indicating success of the preceding operation.

```
HTTP/1.1 200 OK
```

```
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodResponse

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLERSP>
         <IMETHODRESPONSE NAME="CreateClass"/>
      </SIMPLERSP>
   </MESSAGE>
</CIM>
```

A.6 Creation of a Single Instance Definition

The following HTTP request illustrates how a client creates an instance of the class MySchema_VideoBIOSElement. For clarity of exposition, most of the submitted <INSTANCE> element is omitted from the example.

```
M-POST /cimom HTTP/1.1
HOST: http://www.myhost.com/
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: CreateInstance
73-CIMObject: root/cimv2

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLEREQ>
         <IMETHODCALL NAME="CreateInstance">
            <LOCALNAMESPACEPATH>
               <NAMESPACE NAME="root"/>
               <NAMESPACE NAME="cimv2"/>
            </LOCALNAMESPACEPATH>
            <IPARAMVALUE NAME="NewInstance">
               <INSTANCE CLASSNAME="CIM_VideoBIOSElement">
                  ...
               </INSTANCE>
            </IPARAMVALUE>
         </IMETHODCALL>
      </SIMPLEREQ>
   </MESSAGE>
</CIM>
```

Following is an HTTP response to the preceding request indicating the success of the preceding operation.

```
HTTP/1.1 200 OK
```

```
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodResponse

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLERSP>
         <IMETHODRESPONSE NAME="CreateInstance">
            <IRETURNVALUE>
               <INSTANCENAME CLASSNAME="MySchema_VideoBIOSElement">
                  <KEYBINDING NAME="Name"><KEYVALUE>S4</KEYVALUE></KEYBINDING>
               </INSTANCENAME>
            </IRETURNVALUE>
         </IRETURNVALUE>
      </SIMPLERSP>
   </MESSAGE>
</CIM>
```

A.7 Enumeration of Class Names

The following HTTP request illustrates how a client enumerates the names of all subclasses of the class CIM_SoftwareElement.

```
M-POST /cimom HTTP/1.1
HOST: http://www.myhost.com/
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: EnumerateClassNames
73-CIMObject: root/cimv2

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLEREQ>
         <IMETHODCALL NAME="EnumerateClassNames">
            <LOCALNAMESPACEPATH>
               <NAMESPACE NAME="root"/>
               <NAMESPACE NAME="cimv2"/>
            </LOCALNAMESPACEPATH>
            <IPARAMVALUE NAME="ClassName">
               <CLASSNAME NAME="CIM_SoftwareElement"/>
            </IPARAMVALUE>
            <IPARAMVALUE NAME="DeepInheritance">
               <VALUE>FALSE</VALUE>
            </IPARAMVALUE>
         </IMETHODCALL>
      </SIMPLEREQ>
```

```
      </MESSAGE>
   </CIM>
```

Following is an HTTP response to the preceding request indicating the success of the preceding operation and returning the names of the requested subclasses.

```
   HTTP/1.1 200 OK
   Content-Type: application/xml; charset=utf-8
   Content-Length: xxxx
   Ext:
   Cache-Control: no-cache
   Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
   73-CIMOperation: MethodResponse

   <?xml version="1.0" encoding="utf-8" ?>
   <CIM CIMVERSION="2.0" DTDVERSION="2.0">
      <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
         <SIMPLERSP>
            <IMETHODRESPONSE NAME="EnumerateClassNames">
               <IRETURNVALUE>
                  <CLASSNAME NAME="CIM_BIOSElement"/>
                  <CLASSNAME NAME="CIM_VideoBOISElement"/>
               </IRETURNVALUE>
            </IMETHODRESPONSE>
         </SIMPLERSP>
      </MESSAGE>
   </CIM>
```

A.8 Enumeration of Instances

The following HTTP request illustrates how a client enumerates all instances of the class CIM_LogicalDisk. For clarity of exposition, most of the returned instances are omitted from the example.

```
   M-POST /cimom HTTP/1.1
   HOST: http://www.myhost.com/
   Content-Type: application/xml; charset=utf-8
   Content-Length: xxxx
   Man: http://www.dmtf.org/cim/operation ; ns=73
   73-CIMOperation: MethodCall
   73-CIMMethod: EnumerateInstances
   73-CIMObject: root/cimv2

   <?xml version="1.0" encoding="utf-8" ?>
   <CIM CIMVERSION="2.0" DTDVERSION="2.0">
      <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
         <SIMPLEREQ>
            <IMETHODCALL NAME="EnumerateInstances">
               <LOCALNAMESPACEPATH>
                  <NAMESPACE NAME="root"/>
                  <NAMESPACE NAME="cimv2"/>
               </LOCALNAMESPACEPATH>
               <IPARAMVALUE NAME="ClassName">
                  <CLASSNAME NAME="CIM_LogicalDisk"/>
               </IPARAMVALUE>
```

```
                <IPARAMVALUE NAME="LocalOnly"><VALUE>TRUE</VALUE></IPARAMVALUE>
                <IPARAMVALUE NAME="DeepInheritance"><VALUE>TRUE</VALUE></IPARAMVALUE>
             </IMETHODCALL>
          </SIMPLEREQ>
       </MESSAGE>
    </CIM>
```

Following is an HTTP response to the preceding request indicating success of the preceding operation, returning the requested instances.

```
    HTTP/1.1 200 OK
    Content-Type: application/xml; charset=utf-8
    Content-Length: xxxx
    Ext:
    Cache-Control: no-cache
    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
    73-CIMOperation: MethodResponse

    <?xml version="1.0" encoding="utf-8" ?>
    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
       <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
          <SIMPLERSP>
             <IMETHODRESPONSE NAME="EnumerateInstances">
                <IRETURNVALUE>
                   <VALUE.NAMEDINSTANCE>
                      <INSTANCENAME CLASSNAME="Erewhon_LogicalDisk">
                         ...
                      </INSTANCENAME>
                      <INSTANCE CLASSNAME="Erewhon_LogicalDisk">
                         ...
                      </INSTANCE>
                   </VALUE.NAMEDINSTANCE>
                   ...
                   <VALUE.NAMEDINSTANCE>
                      <INSTANCENAME CLASSNAME="Foobar_LogicalDisk">
                         ...
                      </INSTANCENAME>
                      <INSTANCE CLASSNAME="Foobar_LogicalDisk">
                         ...
                      </INSTANCE>
                   </VALUE.NAMEINSTANCE>
                </IRETURNVALUE>
             </IMETHODRESPONSE>
          </SIMPLERSP>
       </MESSAGE>
    </CIM>
```

A.9 Retrieval of a Single Property

The following HTTP request illustrates how a client retrieves the FreeSpace property from the instance MyDisk.DeviceID="C:". This example demonstrates how to use the GetInstance operation with a property list filter instead of the deprecated GetProperty operation.

```
    M-POST /cimom HTTP/1.1
```

```
HOST: http://www.myhost.com/
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Man: http://www.dmtf.org/cim/operation ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: GetInstance
73-CIMObject: root%2FmyNamespace

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLEREQ>
         <IMETHODCALL NAME="GetInstance">
            <LOCALNAMESPACEPATH>
               <NAMESPACE NAME="root"/>
               <NAMESPACE NAME="myNamespace"/>
            </LOCALNAMESPACEPATH>
            <IPARAMVALUE NAME="InstanceName">
               <INSTANCENAME CLASSNAME="MyDisk">
                  <KEYBINDING NAME="DeviceID">
                     <KEYVALUE>C:</KEYVALUE>
                  </KEYBINDING>
               </INSTANCENAME>
            </IPARAMVALUE>
            <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
            <IPARAMVALUE NAME="PropertyList">
               <VALUE>FreeSpace</VALUE>
             </IPARAMVALUE>
         </IMETHODCALL>
      </SIMPLEREQ>
   </MESSAGE>
</CIM>
```

Following is an HTTP response to the preceding request indicating success of the preceding operation, returning the requested instance with the requested property value.

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodResponse

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLERSP>
         <IMETHODRESPONSE NAME="GetInstance">
            <IRETURNVALUE>
               <INSTANCE CLASSNAME="Erewhon_LogicalDisk">
                  <PROPERTY NAME="FreeSpace" TYPE="uint32">
                     <VALUE>6752332</VALUE>
                  </PROPERTY>
```

```
                    </INSTANCE>
                </IRETURNVALUE>
            </IMETHODRESPONSE>
        </SIMPLERSP>
    </MESSAGE>
</CIM>
```

A.10 Execution of an Extrinsic Method

The following HTTP request illustrates how a client executes the SetPowerState method on the instance MyDisk.DeviceID="C:".

```
M-POST /cimom HTTP/1.1
HOST: http://www.myhost.com/
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: SetPowerState
73-CIMObject: root%2FmyNamespace%3AMyDisk.Name%3D%22C%3A%22

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLEREQ>
         <METHODCALL NAME="SetPowerState">
            <LOCALINSTANCEPATH>
               <LOCALNAMESPACEPATH>
                  <NAMESPACE NAME="root"/>
                  <NAMESPACE NAME="myNamespace"/>
               </LOCALNAMESPACEPATH>
               <INSTANCENAME CLASSNAME="MyDisk">
                  <KEYBINDING NAME="Name"><KEYVALUE>C:</KEYVALUE></KEYBINDING>
               </INSTANCENAME>
            </LOCALINSTANCEPATH>
            <PARAMVALUE NAME="PowerState"><VALUE>1</VALUE></PARAMVALUE>
            <PARAMVALUE NAME="Time">
               <VALUE>00000001132312.000000:000</VALUE>
            </PARAMVALUE>
         </METHODCALL>
      </SIMPLEREQ>
   </MESSAGE>
</CIM>
```

Following is an HTTP response to the preceding request indicating the success of the preceding operation.

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodResponse
```

```
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
      <SIMPLERSP>
         <METHODRESPONSE NAME="SetPowerState">
            <RETURNVALUE>
               <VALUE>0</VALUE>
            </RETURNVALUE>
         </METHODRESPONSE>
      </SIMPLERSP>
   </MESSAGE>
</CIM>
```

A.11 Indication Delivery Example

The following HTTP request illustrates the format for sending an indication of type CIM_AlertIndication to a WBEM listener.

```
M-POST /cimlistener/browser HTTP/1.1
HOST: http://www.acme.com/
Content-Type: application/xml; charset=utf-8
Content-Length: XXX
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=40
40-CIMExport: MethodRequest
40-CIMExportMethod: ExportIndication

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="1007" PROTOCOLVERSION="1.0">
      <SIMPLEEXPREQ>
         <EXPMETHODCALL NAME="ExportIndication">
            <EXPPARAMVALUE NAME="NewIndication">
               <INSTANCE CLASSNAME="CIM_AlertIndication" >
                  <PROPERTY NAME="Description" TYPE="string">
                     <VALUE>Sample CIM_AlertIndication indication</VALUE>
                  </PROPERTY>
                  <PROPERTY NAME="AlertType" TYPE="uint16">
                     <VALUE>1</VALUE>
                  </PROPERTY>
                  <PROPERTY NAME="PerceivedSeverity" TYPE="uint16">
                     <VALUE>3</VALUE>
                  </PROPERTY>
                  <PROPERTY NAME="ProbableCause" TYPE="uint16">
                     <VALUE>2</VALUE>
                  </PROPERTY>
                  <PROPERTY NAME="IndicationTime" TYPE="datetime">
                     <VALUE>20010515104354.000000:000</VALUE>
                  </PROPERTY>
               </INSTANCE>
            </EXPPARAMVALUE>
         </EXPMETHODCALL>
      </SIMPLEEXPREQ>
```

```
      </MESSAGE>
   </CIM>
```

Following is an HTTP response to the preceding request indicating a successful receipt by the WBEM listener.

```
   HTTP/1.1 200 OK
   Content-Type: application/xml; charset=utf-8
   Content-Length: 267
   Ext:
   Cache-Control: no-cache
   Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=40
   40-CIMExport: MethodResponse

   <?xml version="1.0" encoding="utf-8" ?>
   <CIM CIMVERSION="2.0" DTDVERSION="2.0">
      <MESSAGE ID="1007" PROTOCOLVERSION="1.0">
         <SIMPLEEXPRSP>
            <EXPMETHODRESPONSE NAME="ExportIndication">
               <IRETURNVALUE></IRETURNVALUE>
            </EXPMETHODRESPONSE>
         </SIMPLEEXPRSP>
      </MESSAGE>
   </CIM>
```

A.12 Subscription Example

A WBEM client application activates a subscription by creating an instance of the CIM_IndicationSubscription class, which defines an association between a CIM_IndicationFilter (a filter) instance and a CIM_IndicationHandler (a handler) instance. The CIM_IndicationFilter instance defines the filter criteria and data project list to describe the desired indication stream. The CIM_IndicationHandler instance defines the desired indication encoding, destination location, and protocol for delivering the indication stream.

The following HTTP request illustrates how a client creates an instance of the class CIM_IndicationFilter. Note that the exact syntax of the WMI Query Language is still under review and is subject to change.

```
   Host: bryce
   Content-Type: application/xml; charset=utf-8
   Content-Length: XXXX
   Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=20
   20-CIMProtocolVersion: 1.0
   20-CIMOperation: MethodCall
   20-CIMMethod: CreateInstance
   20-CIMObject: root/cimv2

   <?xml version="1.0" encoding="utf-8"?>
   <CIM CIMVERSION="2.0" DTDVERSION="2.0">
      <MESSAGE ID="53000" PROTOCOLVERSION="1.0">
         <SIMPLEREQ>
            <IMETHODCALL NAME="CreateInstance">
               <LOCALNAMESPACEPATH>
                  <NAMESPACE NAME="root"/>
                  <NAMESPACE NAME="cimv2"/>
               </LOCALNAMESPACEPATH>
```

```
                <IPARAMVALUE NAME="NewInstance">
                   <INSTANCE CLASSNAME="CIM_IndicationFilter">
                      <PROPERTY NAME="SystemCreationClassName" TYPE="string">
                         <VALUE>CIM_UnitaryComputerSystem</VALUE>
                      </PROPERTY>
                      <PROPERTY NAME="SystemName" TYPE="string">
                         <VALUE>server001.acme.com</VALUE>
                      </PROPERTY>
                      <PROPERTY NAME="CreationClassName" TYPE="string">
                         <VALUE>CIM_IndicationFilter</VALUE>
                      </PROPERTY>
                      <PROPERTY NAME="Name" TYPE="string">
                         <VALUE>ACMESubscription12345</VALUE>
                      </PROPERTY>
                      <PROPERTY NAME="SourceNamespace" TYPE="string">
                         <VALUE>root/cimv2</VALUE>
                      </PROPERTY>
                      <PROPERTY NAME="Query" TYPE="string">
                         <VALUE>
                         SELECT Description, AlertType, PerceivedSeverity,
                             ProbableCause, IndicationTime
                             FROM CIM_AlertIndication
                             WHERE PerceivedSeverity = 3
                         </VALUE>
                      </PROPERTY>
                      <PROPERTY NAME="QueryLanguage" TYPE="string">
                         <VALUE>WQL</VALUE>
                      </PROPERTY>
                   </INSTANCE>
                </IPARAMVALUE>
             </IMETHODCALL>
          </SIMPLEREQ>
       </MESSAGE>
    </CIM>
```

Following is an HTTP response to the preceding request indicating success of the preceding operation.

```
    HTTP/1.1 200 OK
    Content-Type: application/xml; charset=utf-8
    Content-Length: XXX
    Ext:
    Cache-Control: no-cache
    Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=28
    28-CIMOperation: MethodResponse

    <?xml version="1.0" encoding="utf-8" ?>
    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
       <MESSAGE ID="53000" PROTOCOLVERSION="1.0">
          <SIMPLERSP>
             <IMETHODRESPONSE NAME="CreateInstance">
                <IRETURNVALUE>
                   <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
                      <KEYBINDING NAME="SystemCreationClassName">
                         <KEYVALUE VALUETYPE="string">
```

```
                        CIM_UnitaryComputerSystem
                     </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="SystemName">
                     <KEYVALUE VALUETYPE="string">
                        server001.acme.com
                     </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="CreationClassName">
                     <KEYVALUE VALUETYPE="string">
                        CIM_IndicationFilter
                     </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="Name">
                     <KEYVALUE VALUETYPE="string">
                        ACMESubscription12345
                     </KEYVALUE>
                  </KEYBINDING>
               </INSTANCENAME>
            </IRETURNVALUE>
         </IMETHODRESPONSE>
      </SIMPLERSP>
   </MESSAGE>
</CIM>
```

The following HTTP request illustrates how a client creates an instance of the class CIM_IndicationHandlerCIMXML.

```
M-POST /cimom HTTP/1.1
Host: bryce
Content-Type: application/xml; charset=utf-8
Content-Length: XXX
Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=20
20-CIMProtocolVersion: 1.0
20-CIMOperation: MethodCall
20-CIMMethod: CreateInstance
20-CIMObject: root/cimv2

<?xml version="1.0" encoding="utf-8"?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="54000" PROTOCOLVERSION="1.0">
      <SIMPLEREQ>
         <IMETHODCALL NAME="CreateInstance">
            <LOCALNAMESPACEPATH>
               <NAMESPACE NAME="root"/>
               <NAMESPACE NAME="cimv2"/>
            </LOCALNAMESPACEPATH>
            <IPARAMVALUE NAME="NewInstance">
               <INSTANCE CLASSNAME="CIM_IndicationHandlerCIMXML">
                  <PROPERTY NAME="SystemCreationClassName" TYPE="string">
                     <VALUE>CIM_UnitaryComputerSystem</VALUE>
                  </PROPERTY>
                  <PROPERTY NAME="SystemName" TYPE="string">
                     <VALUE>server001.acme.com</VALUE>
```

```
                        </PROPERTY>
                        <PROPERTY NAME="CreationClassName" TYPE="string">
                           <VALUE>CIM_IndicationHandlerCIMXML</VALUE>
                        </PROPERTY>
                        <PROPERTY NAME="Name" TYPE="string">
                           <VALUE>ACMESubscription12345</VALUE>
                        </PROPERTY>
                        <PROPERTY NAME="Owner" TYPE="string">
                           <VALUE>ACMEAlertMonitoringConsole</VALUE>
                        </PROPERTY>
                        <PROPERTY NAME="Destination" TYPE="string">
                           <VALUE>HTTP://www.acme.com/cimlistener/browser</VALUE>
                        </PROPERTY>
                     </INSTANCE>
                  </IPARAMVALUE>
               </IMETHODCALL>
            </SIMPLEREQ>
         </MESSAGE>
      </CIM>
```

Following is an HTTP response to the preceding request indicating the success of the preceding operation.

```
      HTTP/1.1 200 OK
      Content-Type: application/xml; charset=utf-8
      Content-Length: XXX
      Ext:
      Cache-Control: no-cache
      Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=27
      27-CIMOperation: MethodResponse

      <?xml version="1.0" encoding="utf-8" ?>
      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
         <MESSAGE ID="54000" PROTOCOLVERSION="1.0">
            <SIMPLERSP>
               <IMETHODRESPONSE NAME="CreateInstance">
                  <IRETURNVALUE>
                     <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
                        <KEYBINDING NAME="SystemCreationClassName">
                           <KEYVALUE VALUETYPE="string">
                              CIM_UnitaryComputerSystem
                           </KEYVALUE>
                        </KEYBINDING>
                        <KEYBINDING NAME="SystemName">
                           <KEYVALUE VALUETYPE="string">
                              server001.acme.com
                           </KEYVALUE>
                        </KEYBINDING>
                        <KEYBINDING NAME="CreationClassName">
                           <KEYVALUE VALUETYPE="string">
                              CIM_IndicationHandlerCIMXML
                           </KEYVALUE>
                        </KEYBINDING>
                        <KEYBINDING NAME="Name">
```

```
                        <KEYVALUE VALUETYPE="string">
                            ACMESubscription12345
                        </KEYVALUE>
                    </KEYBINDING>
                </INSTANCENAME>
            </IRETURNVALUE>
        </IMETHODRESPONSE>
    </SIMPLERSP>
  </MESSAGE>
</CIM>
```

The following HTTP request illustrates how a client creates an instance of the class CIM_IndicationSubscription.

```
M-POST /cimom HTTP/1.1
Host: bryce
Content-Type: application/xml; charset=utf-8
Content-Length: XXXX
Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=55
55-CIMProtocolVersion: 1.0
55-CIMOperation: MethodCall
55-CIMMethod: CreateInstance
55-CIMObject: root/cimv2

<?xml version="1.0" encoding="utf-8"?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="55000" PROTOCOLVERSION="1.0">
    <SIMPLEREQ>
      <IMETHODCALL NAME="CreateInstance">
        <LOCALNAMESPACEPATH>
          <NAMESPACE NAME="root"/>
          <NAMESPACE NAME="cimv2"/>
        </LOCALNAMESPACEPATH>
        <IPARAMVALUE NAME="NewInstance">
          <INSTANCE CLASSNAME="CIM_IndicationSubscription">
            <PROPERTY.REFERENCE NAME="Filter"
                                REFERENCECLASS="CIM_IndicationFilter">
              <VALUE.REFERENCE>
                <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
                  <KEYBINDING NAME="SystemCreationClassName">
                    <KEYVALUE VALUETYPE="string">
                      CIM_UnitaryComputerSystem
                    </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="SystemName">
                    <KEYVALUE VALUETYPE="string">
                      server001.acme.com
                    </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="CreationClassName">
                    <KEYVALUE VALUETYPE="string">
                      CIM_IndicationFilter
                    </KEYVALUE>
                  </KEYBINDING>
```

```
                                    <KEYBINDING NAME="Name">
                                        <KEYVALUE VALUETYPE="string">
                                            ACMESubscription12345
                                        </KEYVALUE>
                                    </KEYBINDING>
                                </INSTANCENAME>
                            </VALUE.REFERENCE>
                        </PROPERTY.REFERENCE>
                        <PROPERTY.REFERENCE NAME="Handler"
                                            REFERENCECLASS="CIM_IndicationHandler">
                            <VALUE.REFERENCE>
                                <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
                                    <KEYBINDING NAME="SystemCreationClassName">
                                        <KEYVALUE VALUETYPE="string">
                                            CIM_UnitaryComputerSystem
                                        </KEYVALUE>
                                    </KEYBINDING>
                                    <KEYBINDING NAME="SystemName">
                                        <KEYVALUE VALUETYPE="string">
                                            server001.acme.com
                                        </KEYVALUE>
                                    </KEYBINDING>
                                    <KEYBINDING NAME="CreationClassName">
                                        <KEYVALUE VALUETYPE="string">
                                            CIM_IndicationHandlerCIMXML
                                        </KEYVALUE>
                                    </KEYBINDING>
                                    <KEYBINDING NAME="Name">
                                        <KEYVALUE VALUETYPE="string">
                                            ACMESubscription12345
                                        </KEYVALUE>
                                    </KEYBINDING>
                                </INSTANCENAME>
                            </VALUE.REFERENCE>
                        </PROPERTY.REFERENCE>
                    </INSTANCE>
                </IPARAMVALUE>
            </IMETHODCALL>
        </SIMPLEREQ>
    </MESSAGE>
</CIM>
```

Following is an HTTP response to the preceding request indicating the success of the preceding operation.

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
Content-Length: XXXX
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=75
75-CIMOperation: MethodResponse

<?xml version="1.0" encoding="utf-8" ?>
```

```
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="55000" PROTOCOLVERSION="1.0">
    <SIMPLERSP>
      <IMETHODRESPONSE NAME="CreateInstance">
        <IRETURNVALUE>
          <INSTANCENAME CLASSNAME="CIM_IndicationSubscription">
            <KEYBINDING NAME="Filter">
              <VALUE.REFERENCE>
                <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
                  <KEYBINDING NAME="SystemCreationClassName">
                    <KEYVALUE VALUETYPE="string">
                      CIM_UnitaryComputerSystem
                    </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="SystemName">
                    <KEYVALUE VALUETYPE="string">
                      server001.acme.com
                    </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="CreationClassName">
                    <KEYVALUE VALUETYPE="string">
                      CIM_IndicationFilter
                    </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="Name">
                    <KEYVALUE VALUETYPE="string">
                      ACMESubscription12345
                    </KEYVALUE>
                  </KEYBINDING>
                </INSTANCENAME>
              </VALUE.REFERENCE>
            </KEYBINDING>
            <KEYBINDING NAME="Handler">
              <VALUE.REFERENCE>
                <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
                  <KEYBINDING NAME="SystemCreationClassName">
                    <KEYVALUE VALUETYPE="string">
                      CIM_UnitaryComputerSystem
                    </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="SystemName">
                    <KEYVALUE VALUETYPE="string">
                     server001.acme.com
                    </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="CreationClassName">
                    <KEYVALUE VALUETYPE="string">
                      CIM_IndicationHandlerCIMXML
                    </KEYVALUE>
                  </KEYBINDING>
                  <KEYBINDING NAME="Name">
                    <KEYVALUE VALUETYPE="string">
                      ACMESubscription12345
```

```
                                    </KEYVALUE>
                                </KEYBINDING>
                            </INSTANCENAME>
                        </VALUE.REFERENCE>
                    </KEYBINDING>
                </INSTANCENAME>
            </IRETURNVALUE>
        </IMETHODRESPONSE>
    </SIMPLERSP>
  </MESSAGE>
</CIM>
```

A.13 Multiple Operations Example

The following HTTP request illustrates how a client performs multiple operations. This example batches a GetClass, an EnumerateInstanceNames, and an EnumerateInstance operation on CIM_ObjectManagerAdapter.

```
POST /CIMOM1 HTTP/1.1
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Content-Length: XXX
Host: localhost:5988
CIMOperation: MethodCall
CIMProtocolVersion: 1.0
Content-Type: application/xml; charset=utf-8
CIMBatch: CIMBatch
<?xml version="1.0" encoding="UTF-8"?>

<CIM DTDVERSION="2.0" CIMVERSION="2.0">
   <MESSAGE ID="2004:2:5:1:1:11:41:1" PROTOCOLVERSION="1.0">
      <MULTIREQ>
         <SIMPLEREQ>
            <IMETHODCALL NAME="GetClass">
               <LOCALNAMESPACEPATH>
                  <NAMESPACE NAME="interop" />
               </LOCALNAMESPACEPATH>
               <IPARAMVALUE NAME="ClassName">
                  <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
               </IPARAMVALUE>
               <IPARAMVALUE NAME="LocalOnly">
                  <VALUE>FALSE</VALUE>
               </IPARAMVALUE>
               <IPARAMVALUE NAME="IncludeClassOrigin">
                  <VALUE>TRUE</VALUE>
               </IPARAMVALUE>
            </IMETHODCALL>
         </SIMPLEREQ>
         <SIMPLEREQ>
            <IMETHODCALL NAME="Associators">
               <LOCALNAMESPACEPATH>
                  <NAMESPACE NAME="interop" />
               </LOCALNAMESPACEPATH>
               <IPARAMVALUE NAME="ObjectName">
                  <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
```

```
                    </IPARAMVALUE>
                    <IPARAMVALUE NAME="IncludeQualifiers">
                       <VALUE>TRUE</VALUE>
                    </IPARAMVALUE>
                    <IPARAMVALUE NAME="IncludeClassOrigin">
                       <VALUE>TRUE</VALUE>
                    </IPARAMVALUE>
                 </IMETHODCALL>
              </SIMPLEREQ>
              <SIMPLEREQ>
                 <IMETHODCALL NAME="EnumerateInstanceNames">
                    <LOCALNAMESPACEPATH>
                       <NAMESPACE NAME="interop" />
                    </LOCALNAMESPACEPATH>
                    <IPARAMVALUE NAME="ClassName">
                       <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
                    </IPARAMVALUE>
                 </IMETHODCALL>
              </SIMPLEREQ>
              <SIMPLEREQ>
                 <IMETHODCALL NAME="EnumerateInstances">
                    <LOCALNAMESPACEPATH>
                       <NAMESPACE NAME="interop" />
                    </LOCALNAMESPACEPATH>
                    <IPARAMVALUE NAME="ClassName">
                       <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
                    </IPARAMVALUE>
                    <IPARAMVALUE NAME="LocalOnly">
                       <VALUE>FALSE</VALUE>
                    </IPARAMVALUE>
                 </IMETHODCALL>
              </SIMPLEREQ>
           </MULTIREQ>
        </MESSAGE>
     </CIM>
```

Following is the HTTP response to the preceding request indicating the success of the preceding operation.

```
     HTTP/1.1 200 OK
     CIMOperation: MethodResponse
     Content-Length: XXX

     <?xml version="1.0" encoding="UTF-8"?>
     <CIM DTDVERSION="2.0" CIMVERSION="2.0">
        <MESSAGE ID="2004:2:5:1:1:11:41:1" PROTOCOLVERSION="1.0">
           <MULTIRSP>
              <SIMPLERSP>
                 <IMETHODRESPONSE NAME="GetClass">
                    <IRETURNVALUE>
                       <CLASS SUPERCLASS="CIM_WBEMService"
                              NAME="CIM_ObjectManagerAdapter">
                          ...
                       </CLASS>
```

```
                    </IRETURNVALUE>
                </IMETHODRESPONSE>
            </SIMPLERSP>
            <SIMPLERSP>
                <IMETHODRESPONSE NAME="Associators">
                    <IRETURNVALUE>
                        <VALUE.OBJECTWITHPATH>
                            ...
                        </VALUE.OBJECTWITHPATH>
                        <VALUE.OBJECTWITHPATH>
                            ...
                        </VALUE.OBJECTWITHPATH>
                            ...
                    </IRETURNVALUE>
                </IMETHODRESPONSE>
            </SIMPLERSP>
            <SIMPLERSP>
                <IMETHODRESPONSE NAME="EnumerateInstanceNames">
                    <IRETURNVALUE>
                        <INSTANCENAME CLASSNAME="WBEMSolutions_ObjectManagerAdapter">
                            ...
                        </INSTANCENAME>
                        <INSTANCENAME CLASSNAME="WBEMSolutions_ObjectManagerAdapter">
                            ...
                        </INSTANCENAME>
                            ...
                    </IRETURNVALUE>
                </IMETHODRESPONSE>
            </SIMPLERSP>
            <SIMPLERSP>
                <IMETHODRESPONSE NAME="EnumerateInstances">
                    <IRETURNVALUE>
                        <VALUE.NAMEDINSTANCE>
                            ...
                        </VALUE.NAMEDINSTANCE>
                        <VALUE.NAMEDINSTANCE>
                            ...
                        </VALUE.NAMEDINSTANCE>
                            ...
                    </IRETURNVALUE>
                </IMETHODRESPONSE>
            </SIMPLERSP>
        </MULTIRSP>
    </MESSAGE>
</CIM>
```

ANNEX B
(informative)

LocalOnly Parameter Discussion

This annex discusses the issues associated with the 1.1 definition of the `LocalOnly` parameter for the GetInstance and EnumerateInstances operations.

B.1 Explanation of the Deprecated 1.1 Interpretation

In April 2002, two DMTF Change Requests (CRs), CR809 (EnumerateInstances) and CR815 (GetInstance), were approved and incorporated into version 1.1of this document to clarify the interpretation of the `LocalOnly` flag for the GetInstance and EnumerateInstances operations. With these CRs, the definition of the `LocalOnly` flag for these operations was modified to align with the interpretation of this flag for the GetClass and EnumerateClasses operations. This change was incorrect, resulted in reduced functionality, and introduced several backward compatibility issues.

To clarify the difference between the 1.0 Interpretation and the 1.1 Interpretation (CR815), consider the following example:

```
class A {
      [Key]
   string name;
   uint32 counter = 3;
};

class B : A {
   uint32 moreData = 4;
};

instance of A {
   name = "Roger";
};

instance of B {
   name = "Karl";
   counter = 3;
   moreData = 5;
};

instance of B {
   name = "Denise";
   counter = 5;
};
```

Assuming `PropertyList = NULL` and `LocalOnly = TRUE`, Table 7 shows the properties returned by a GetInstance operation.

Table 7 – Comparison of Properties Returned by GetInstance in Versions 1.0 and 1.1

Instance	DSP0200 1.0 Interpretation	DSP0200 1.1 Interpretation
"Roger"	name	name, counter
"Karl"	name, counter, moreData	moreData
"Denise"	name, counter	moreData

The properties returned using the 1.0 interpretation are consistent with the properties specified in the MOF instance definitions, and the properties returned using the 1.1 Interpretation are consistent with the properties defined in the class definitions.

B.2 Risks of Using the 1.1 Interpretation

The risks of using the 1.1 interpretation are as follows:

1) Within the DMTF, promoting a property from a class to one of its superclasses is defined as a backward-compatible change that can be made in a minor revision of the CIM schema. With the 1.1 interpretation, promoting a property to a superclass can cause backward-incompatible changes.

   Suppose, for example, version 1.0 of the schema includes the following definitions:

```
class A {
      [Key]
   string name;
   uint32 counter = 3;
};

class B : A {
   uint32 moreData = 4;
};
```

   Now suppose that the schema is modified in version 1.1 to promote the property moreData from class B to class A.

```
class A {
      [Key]
   string name;
   uint32 counter = 3;
   uint32 moreData = 4;
};

class B : A {
};
```

   Using these examples, Table 8 shows the properties returned by a call to GetInstance with `PropertyList = NULL` and `LocalOnly = TRUE`. With the 1.1 Interpretation, this schema change would affect the list of properties returned. When dealing with a WBEM server that complies with the 1.1 interpretation, applications must be designed to treat “promoting properties” as a backward-compatible change.

4971 **Table 8 – Comparison of Properties Returned by a Call to GetInstance in Versions 1.0 and 1.1**

Instance	Schema Version 1.0	Schema Version 1.1
of A	name, counter	name, counter, moreData
of B	moreData	none

4972

4973 2) The 1.1 Interpretation encourages application developers to use multiple operations to retrieve the
4974 properties of an instance. That is, a commonly-stated use model for the 1.1 interpretation is to
4975 selectively traverse subclasses getting additional properties of an instance. This practice significantly
4976 increases the risk that a client will construct an inconsistent instance. With both Interpretations,
4977 applications should be designed to ensure that dependent properties are retrieved together.

4978 B.3 Techniques for Differentiating between the 1.0 Interpretation and 1.1 4979 Interpretation

4980 For concrete classes, WBEM servers that comply with the 1.0 Interpretation return the value of all KEY
4981 properties not explicitly excluded by the `PropertyList` parameter. WBEM servers that comply with the
4982 1.1 interpretation return only the value of KEY properties explicitly defined in the class. Applications can
4983 use this difference to detect which interpretation is supported by a WBEM server.

ANNEX C
(normative)

Generic Operations Mapping

This annex defines a mapping of generic operations (see DSP0223) to the CIM-XML protocol described in this document.

A main purpose of this mapping is to support the implementations of DMTF management profiles that define operations in terms of generic operations, by providing them a translation from the generic operation listed in the management profile, to the CIM-XML operation that actually needs to be implemented.

C.1 Operations

This subclause defines for each generic operation, which CIM-XML operation needs to be supported in order to support the respective generic operation.

Table 9 lists the generic operations defined in DSP0223 and for each of them, lists the name of the corresponding CIM-XML operation and a link to the description subclause.

Table 9 – Mapping of generic operations to CIM-XML operations

Generic Operation	CIM-XML Operation	Description
GetInstance	GetInstance	See C.1.1
DeleteInstance	DeleteInstance	See C.1.2
ModifyInstance	ModifyInstance	See C.1.3
CreateInstance	CreateInstance	See C.1.4
GetClassInstancesWithPath	EnumerateInstances	See C.1.5
GetClassInstancePaths	EnumerateInstanceNames	See C.1.6
GetAssociatedInstancesWithPath	Associators (ObjectName is an instance path)	See C.1.7
GetAssociatedInstancePaths	AssociatorNames (ObjectName is an instance path)	See C.1.8
GetReferencingInstancesWithPath	References (ObjectName is an instance path)	See C.1.9
GetReferencingInstancePaths	ReferenceNames (ObjectName is an instance path)	See C.1.10
OpenClassInstancesWithPath	OpenEnumerateInstances	See C.1.11
OpenClassInstancePaths	OpenEnumerateInstancePaths	See C.1.12
OpenAssociatedInstancesWithPath	OpenAssociatorInstances	See C.1.13
OpenAssociatedInstancePaths	OpenAssociatorInstanceNames	See C.1.14
OpenReferencingInstancesWithPath	OpenReferenceInstances	See C.1.15
OpenReferencingInstancePaths	OpenReferenceInstanceNames	See C.1.16
OpenQueryInstances	OpenQueryInstances	See C.1.17
PullInstancesWithPath	PullInstancesWithPath	See C.1.18
PullInstancePaths	PullInstancePaths	See C.1.19

Generic Operation	CIM-XML Operation	Description
PullInstances	PullInstances	See C.1.20
CloseEnumeration	CloseEnumeration	See C.1.21
EnumerationCount	EnumerationCount	See C.1.22
InvokeMethod	invocation of extrinsic non-static method	See C.1.23
InvokeStaticMethod	invocation of extrinsic static method	See C.1.24
GetClass	GetClass	See C.1.25
DeleteClass	DeleteClass	See C.1.26
ModifyClass	ModifyClass	See C.1.27
CreateClass	CreateClass	See C.1.28
GetTopClassesWithPath	EnumerateClasses (ClassName is NULL)	See C.1.29
GetTopClassPaths	EnumerateClassNames (ClassName is NULL)	See C.1.30
GetSubClassesWithPath	EnumerateClasses (ClassName is non-NULL)	See C.1.31
GetSubClassPaths	EnumerateClassNames (ClassName is non-NULL)	See C.1.32
GetAssociatedClassesWithPath	Associators (ObjectName is a class path)	See C.1.33
GetAssociatedClassPaths	AssociatorNames (ObjectName is a class path)	See C.1.34
GetReferencingClassesWithPath	References (ObjectName is a class path)	See C.1.35
GetReferencingClassPaths	ReferenceNames (ObjectName is a class path)	See C.1.36
GetQualifierType	GetQualifier	See C.1.37
DeleteQualifierType	DeleteQualifier	See C.1.38
ModifyQualifierType	SetQualifier (Qualifier exists)	See C.1.39
CreateQualifierType	SetQualifier (Qualifier does not exist)	See C.1.40
EnumerateQualifierTypesWithPath	EnumerateQualifiers	See C.1.41

5001 In the following subclauses, the CIM-XML Type listed in the tables is either an intrinsic CIM type (e.g.
5002 "boolan"), or one of the pseudo-types defined in this document (e.g. "instanceName").

5003 ### C.1.1 GetInstance

5004 **CIM-XML Operation Name:** GetInstance

5005 **Purpose:** Retrieve an instance given its instance path.

5006 **Operation Input Parameters:**

5007

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
N/A	N/A	IncludeQualifiers	boolean	See 2)
N/A	N/A	LocalOnly	boolean	See 3)

1) The CIM-XML parameter *InstanceName* includes the model path portion of the instance path of the instance. The generic parameter *InstancePath* corresponds to the combination of the CIM-XML parameter *InstanceName* and the target namespace of the CIM-XML operation.

2) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The defined behavior of generic operation *GetInstance* conforms to the behavior of CIM-XML operation *GetInstance* with *IncludeQualifiers=false*, which is the recommended value to be used for CIM-XML clients since version 1.2 of this document.

3) The CIM-XML parameter *LocalOnly* has been deprecated in version 1.2 of this document. The defined behavior of generic operation *GetInstance* conforms to the behavior of CIM-XML operation *GetInstance* with *LocalOnly=false*, which is the recommended value to be used for CIM-XML clients since version 1.2 of this document.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
Instance	InstanceSpecification	return value	instance	

Optional behavior:

- CIM-XML allows implementations to optimize by not including properties in the returned instance that have a value of NULL.

Deviations: None

C.1.2 DeleteInstance

CIM-XML Operation Name: DeleteInstance

Purpose: Delete an instance given its instance path.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)

1) The CIM-XML parameter *InstanceName* includes the model path portion of the instance path of the instance. The generic parameter *InstancePath* corresponds to the combination of the CIM-XML parameter *InstanceName* and the target namespace of the CIM-XML operation.

Operation Output Parameters: None

Deviations: None

C.1.3 ModifyInstance

CIM-XML Operation Name: ModifyInstance

Purpose: Modify property values of an instance given its instance path.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	target namespace	N/A	See 1)
		ModifiedInstance	namedInstance	See 1)
ModifiedInstance	InstanceSpecification	ModifiedInstance	namedInstance	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
N/A	N/A	IncludeQualifiers	boolean	See 2)

1) The CIM-XML parameter *ModifiedInstance* includes the model path portion of the instance path of the instance that is being modified, and the modified property values. The combination of the model path portion of the CIM-XML parameter *ModifiedInstance* and the target namespace of the CIM-XML operation corresponds to the generic parameter *InstancePath*.

2) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The defined behavior of generic operation *ModifyInstance* conforms to the behavior of CIM-XML operation *ModifyInstance* with *IncludeQualifiers=false*, which is the recommended behavior for CIM-XML servers since version 1.2 of this document.

Operation Output Parameters: None

Optional behavior:

- DSP0223 permits conformant WBEM protocols to require that all properties exposed by the creation class of the instance referenced by *InstancePath* are supplied by the WBEM client with their modified values. CIM-XML does not require that, i.e. CIM-XML permits clients to supply modified values only for a subset of these properties and those not supplied are meant to be left unchanged by the operation.

Deviations: None

C.1.4 CreateInstance

CIM-XML Operation Name: CreateInstance

Purpose: Create a CIM instance given the class path of its creation class.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		NewInstance	instance	See 1)
NewInstance	InstanceSpecification	NewInstance	instance	

1) The generic parameter *ClassPath* corresponds to the combination of the class name specified in the CIM-XML parameter *NewInstance* and the target namespace of the CIM-XML operation.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	return value	instanceName	

Optional behavior: None

Deviations: None

C.1.5 GetClassInstancesWithPath

CIM-XML Operation Name: EnumerateInstances

Purpose: Retrieve the instances of a given class (including instances of its subclasses). The retrieved instances include their instance paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	DeepInheritance	boolean	See 2)
N/A	N/A	IncludeQualifiers	boolean	See 3)
N/A	N/A	LocalOnly	boolean	See 4)

1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of the CIM-XML operation.

2) The generic parameter *ExcludeSubclassProperties* corresponds to the negated CIM-XML parameter *DeepInheritance*.

3) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The defined behavior of generic operation *GetClassInstancesWithPath* conforms to the behavior of CIM-XML operation *EnumerateInstances* with *IncludeQualifiers=false*, which is the recommended value to be used for CIM-XML clients since version 1.2 of this document.

4) The CIM-XML parameter *LocalOnly* has been deprecated in version 1.2 of this document. The defined behavior of generic operation *GetClassInstancesWithPath* conforms to the behavior of CIM-XML operation *EnumerateInstances* with *LocalOnly=false*, which is the recommended value to be used for CIM-XML clients since version 1.2 of this document.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	namedInstance [ ]	See 1)

1) The CIM-XML return value includes the set of property values including the model paths, but without namespace paths. The generic parameter *InstanceList* needs to contain the instance paths in addition to the set of property values. A CIM client side mapping layer can construct the instance paths from the model paths and the CIM-XML target namespace.

Optional behavior:

- CIM-XML allows implementations to optimize by not including properties in the returned instances that have a value of NULL.

Deviations: None

C.1.6 GetClassInstancePaths

CIM-XML Operation Name: EnumerateInstanceNames

Purpose: Retrieve the instance paths of the instances of a given class (including instances of its subclasses).

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)

1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of the CIM-XML operation.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instanceName [ ]	See 1)

1) The CIM-XML return value includes the set of model paths, but without namespace paths. The generic parameter *InstancePathList* needs to contain the instance paths, including namespace paths. A CIM client side mapping layer can construct the instance paths from the model paths and the CIM-XML target namespace.

Optional behavior: None

Deviations: None

C.1.7 GetAssociatedInstancesWithPath

CIM-XML Operation Name: Associators with ObjectName being an instance path

Purpose: Retrieve the instances that are associated with a given source instance. The retrieved instances include their instance paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	N/A	N/A	See 2)
N/A	N/A	IncludeQualifiers	boolean	See 3)

1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML parameter *ObjectName* and the target namespace of the CIM-XML operation.

   The generic operation *GetAssociatedInstancesWithPath* corresponds to the CIM-XML operation *Associators* when an instance path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *Associators* with a class path for its *ObjectName* parameter is covered by the generic operation *GetAssociatedClassesWithPath* (see C.1.33).

2) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-XML parameter. Since the defined behavior of the CIM-XML operation will result in including subclass properties, a mapping layer on the CIM client side can implement the behavior defined by the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that parameter has a value of true.

3) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The defined behavior of generic operation *GetAssociatedInstancesWithPath* conforms to the behavior of CIM-XML operation *Associators* with *IncludeQualifiers=false*, which is the recommended value to be used for CIM-XML clients since version 1.2 of this document.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	objectWithPath [ ]	

Optional behavior:

- CIM-XML allows implementations to optimize by not including properties in the returned instances that have a value of NULL.

Deviations: None

C.1.8 GetAssociatedInstancePaths

CIM-XML Operation Name: AssociatorNames with ObjectName being an instance path

Purpose: Retrieve the instance paths of the instances that are associated with a given source instance.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	

1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML parameter *ObjectName* and the target namespace of the CIM-XML operation.

   The generic operation *GetAssociatedInstancePaths* corresponds to the CIM-XML operation *AssociatorNames* when an instance path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *AssociatorNames* with a class path for its *ObjectName* parameter is covered by the generic operation *GetAssociatedClassPaths* (see C.1.34).

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	objectPath [ ]	

Optional behavior: None

Deviations: None

C.1.9 GetReferencingInstancesWithPath

CIM-XML Operation Name: References with ObjectName being an instance path

Purpose: Retrieve the association instances that reference a given source instance. The retrieved instances include their instance paths.

5157 **Operation Input Parameters:**

5158

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See **Error! Reference source not found.**
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See **Error! Reference source not found.**
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	N/A	N/A	See 3)
N/A	N/A	IncludeQualifiers	boolean	See 4)

5159 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML
5160 parameter *ObjectName* and the target namespace of the CIM-XML operation.

5161 The generic operation *GetReferencingInstancesWithPath* corresponds to the CIM-XML operation
5162 *References* when an instance path is passed in for its *ObjectName* parameter. Using the CIM-XML
5163 operation *References* with a class path for its *ObjectName* parameter is covered by the generic
5164 operation *GetReferencingClassesWithPath* (see C.1.35).

5165 2) The CIM-XML operation *References* does not support a means to filter by class name or role name
5166 of the associated classes on the other ends of the associations referencing the source instance. The
5167 generic operation *GetReferencingInstancesWithPat*h does support such filtering through its
5168 parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the
5169 CIM-XML operation will result in including association instances that these two parameters could
5170 filter out, a mapping layer on the CIM client side can implement the behavior defined by these two
5171 generic parameters by eliminating association instances if these filter parameters are used.

5172 3) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-
5173 XML parameter. Since the defined behavior of the CIM-XML operation will result in including
5174 subclass properties, a mapping layer on the CIM client side can implement the behavior defined by
5175 the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that
5176 parameter has a value of true.

5177 4) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The
5178 defined behavior of generic operation *GetReferencingInstancesWithPath* conforms to the behavior of
5179 CIM-XML operation *References* with *IncludeQualifiers=false*, which is the recommended value to be
5180 used for CIM-XML clients since in version 1.2 of this document.

5181 **Operation Output Parameters:**

5182

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	objectWithPath [ ]	

Optional behavior:

- CIM-XML allows implementations to optimize by not including properties in the returned instances that have a value of NULL.

Deviations: None

C.1.10 GetReferencingInstancePaths

CIM-XML Operation Name: ReferenceNames with ObjectName being an instance path

Purpose: Retrieve the instance paths of the association instances that reference a given source instance.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See **Error! Reference source not found.**1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See 2)
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)

1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML parameter *ObjectName* and the target namespace of the CIM-XML operation.

   The generic operation *GetReferencingInstancePaths* corresponds to the CIM-XML operation *ReferenceNames* when an instance path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *ReferenceNames* with a class path for its *ObjectName* parameter is covered by the generic operation *GetReferencingClassPaths* (see C.1.36).

2) The CIM-XML operation *References* does not support a means to filter by class name or role name of the associated classes on the other ends of the associations referencing the source instance. The generic operation *GetReferencingInstancesWithPat*h does support such filtering through its parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the CIM-XML operation will result in including association instances that these two parameters could filter out, a mapping layer on the CIM client side can implement the behavior defined by these two generic parameters by eliminating association instances if these filter parameters are used.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	objectPath [ ]	

Optional behavior: None

Deviations: None

C.1.11 OpenClassInstancesWithPath

CIM-XML Operation Name: OpenEnumerateInstances

Purpose: Open an enumeration session for retrieving the instances of a class (including instances of its subclasses), and optionally retrieve a first set of those instances. The retrieved instances include their instance paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	N/A	N/A	See **Error! Reference source not found.**2)
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of the CIM-XML operation.

2) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-XML parameter. Since the defined behavior of the CIM-XML operation will result in including subclass properties, a mapping layer on the CIM client side can implement the behavior defined by the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that parameter has a value of true.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	instanceWithPath [ ]	

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

Optional behavior:

- CIM-XML allows implementations to optimize by not including properties in the returned instances that have a value of NULL.

Deviations: None

C.1.12 OpenClassInstancePaths

CIM-XML Operation Name: OpenEnumerateInstancePaths

Purpose: Open an enumeration session for retrieving the instance paths of the instances of a class (including instances of its subclasses), and optionally retrieve a first set of those instance paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of the CIM-XML operation.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

Optional behavior: None

Deviations: None

C.1.13 OpenAssociatedInstancesWithPath

CIM-XML Operation Name: OpenAssociatorInstances

Purpose: Open an enumeration session for retrieving the instances that are associated with a given source instance, and optionally retrieve a first set of those instances. The retrieved instances include their instance paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	N/A	N/A	See 2)
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML parameter *InstanceName* and the target namespace of the CIM-XML operation.

2) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-XML parameter. Since the defined behavior of the CIM-XML operation will result in including subclass properties, a mapping layer on the CIM client side can implement the behavior defined by the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that parameter has a value of true.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	instanceWithPath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5258 **Optional behavior:**

5259 • CIM-XML allows implementations to optimize by not including properties in the returned
5260 instances that have a value of NULL.

5261 **Deviations:** None

5262 ### C.1.14 OpenAssociatedInstancePaths

5263 **CIM-XML Operation Name:** OpenAssociatorInstancePaths

5264 **Purpose:** Open an enumeration session for retrieving the instance paths of instances that are associated
5265 with a given source instance, and optionally retrieve a first set of those instance paths.

5266 **Operation Input Parameters:**

5267

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

5268 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML
5269 parameter *InstanceName* and the target namespace of the CIM-XML operation.

5270 **Operation Output Parameters:**

5271

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5272 **Optional behavior:** None

5273 **Deviations:** None

C.1.15 OpenReferencingInstancesWithPath

CIM-XML Operation Name: OpenReferenceInstances

Purpose: Open an enumeration session for retrieving the association instances that reference a given source instance, and optionally retrieve a first set of those instances. The retrieved instances include their instance paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See 2)
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
ExcludeSubclass-Properties	boolean	N/A	N/A	See 3)
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML parameter *InstanceName* and the target namespace of the CIM-XML operation.

2) The CIM-XML operation *OpenReferenceInstances* does not support a means to filter by class name or role name of the associated classes on the other ends of the associations referencing the source instance. The generic operation *OpenReferencingInstancesWithPath* does support such filtering through its parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the CIM-XML operation will result in including association instances that these two parameters could filter out, a mapping layer on the CIM client side can implement the behavior defined by these two generic parameters by eliminating association instances if these filter parameters are used.

3) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-XML parameter. Since the defined behavior of the CIM-XML operation will result in including subclass properties, a mapping layer on the CIM client side can implement the behavior defined by the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that parameter has a value of true.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	instanceWithPath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

Optional behavior:

- CIM-XML allows implementations to optimize by not including properties in the returned instances that have a value of NULL.

Deviations: None

C.1.16 OpenReferencingInstancePaths

CIM-XML Operation Name: OpenReferenceInstancePaths

Purpose: Open an enumeration session for retrieving the instance paths of association instances that reference a given source instance, and optionally retrieve a first set of those instance paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
SourceInstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See 2)
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)
FilterQueryString	QueryString	FilterQuery	string	
FilterQueryLanguage	QueryLanguage	FilterQueryLanguage	string	
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML parameter *InstanceName* and the target namespace of the CIM-XML operation.

2) The CIM-XML operation *OpenReferenceInstancePaths* does not support a means to filter by class name or role name of the associated classes on the other ends of the associations referencing the source instance. The generic operation *OpenReferencingInstancePaths* does support such filtering through its parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the CIM-XML operation will result in including association instances that these two parameters could filter out, a mapping layer on the CIM client side can implement the behavior defined by these two generic parameters by eliminating association instances if these filter parameters are used.

5316 **Operation Output Parameters:**

5317

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5318 **Optional behavior:** None

5319 **Deviations:** None

5320 ### C.1.17 OpenQueryInstances

5321 **CIM-XML Operation Name:** OpenQueryInstances

5322 **Purpose:** Open an enumeration session for retrieving the instances representing a query result, and
5323 optionally retrieve a first set of those instances. The retrieved instances are not addressable and thus do
5324 not include any instance paths.

5325 **Operation Input Parameters:**

5326

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
QueryString	QueryString	FilterQuery	string	
QueryLanguage	QueryLanguage	FilterQueryLanguage	string	
ReturnQueryResult-Class	boolean	ReturnQueryResult-Class	boolean	
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

5327 **Operation Output Parameters:**

5328

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification [ ]	return value	instance [ ]	
QueryResultClass		QueryResultClass	class	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5329 **Optional behavior:**

- CIM-XML allows implementations to optimize by not including properties in the returned instances that have a value of NULL.

Deviations: None

C.1.18 PullInstancesWithPath

CIM-XML Operation Name: PullInstancesWithPath

Purpose: Retrieve the next set of instances from an open enumeration session. The retrieved instances include their instance paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
MaxObjectCount	uint32	MaxObjectCount	uint32	

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	instanceWithPath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

Optional behavior:

- CIM-XML allows implementations to optimize by not including properties in the returned instances that have a value of NULL.

Deviations: None

C.1.19 PullInstancePaths

CIM-XML Operation Name: PullInstancePaths

Purpose: Retrieve the next set of instance paths from an open enumeration session.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
MaxObjectCount	uint32	MaxObjectCount	uint32	

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

Optional behavior: None

Deviations: None

C.1.20 PullInstances

CIM-XML Operation Name: PullInstances

Purpose: Retrieve the next set of instances from an open enumeration session. The retrieved instances do not include any instance paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
MaxObjectCount	uint32	MaxObjectCount	uint32	

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification [ ]	return value	instance [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

Optional behavior:

- CIM-XML allows implementations to optimize by not including properties in the returned instances that have a value of NULL.

Deviations: None

C.1.21 CloseEnumeration

CIM-XML Operation Name: CloseEnumeration

Purpose: Close an open enumeration session.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	

Operation Output Parameters: None

Optional behavior: None

Deviations: None

C.1.22 EnumerationCount

CIM-XML Operation Name: EnumerationCount

Purpose: Estimate the total number of remaining items in an open enumeration session.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumerationCount	uint64	return value	uint64	

Optional behavior: None

Deviations: None

C.1.23 InvokeMethod

CIM-XML Operation Name: The generic operation *InvokeMethod* corresponds to CIM-XML extrinsic method invocation on an instance. CIM-XML extrinsic method invocation on a class is covered by the generic operation *InvokeStaticMethod* (see C.1.24).

Purpose: Invoke a method on an instance.

Operation Input Parameters:

This document does not define an operation name or parameters for extrinsic method invocation. DSP0201 defines the input and output parameters for extrinsic method invocation by means of the attributes and child elements of the XML elements METHODCALL and METHODRESPONSE. The table below therefore uses the names of these attributes and child elements in the mapping to generic operation parameters.

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	target namespace	N/A	See 1)
		LOCALINSTANCE-PATH child element	N/A	See 1)
MethodName	MethodName	NAME attribute	N/A	
InParmValues	ParameterValue [ ]	set of PARAMVALUE child elements	N/A	

1) The CIM-XML element *LOCALINSTANCEPATH* includes the model path portion of the instance path of the instance. The generic parameter *InstancePath* corresponds to the combination of the CIM-XML element *LOCALINSTANCEPATH* and the target namespace of the CIM-XML operation.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
OutParmValues	ParameterValue [ ]	set of PARAMVALUE child elements	N/A	
ReturnValue	ReturnValue	RETURNVALUE child element	N/A	

Optional behavior: None

Deviations: None

C.1.24 InvokeStaticMethod

CIM-XML Operation Name: The generic operation *InvokeStaticMethod* corresponds to CIM-XML extrinsic method invocation on a class. CIM-XML extrinsic method invocation on an instance is covered by the generic operation *InvokeMethod* (see C.1.23).

Purpose: Invoke a static method on a class.

Operation Input Parameters:

This document does not define an operation name or parameters for extrinsic method invocation. DSP0201 defines the input and output parameters for extrinsic method invocation by means of the attributes and child elements of the XML elements METHODCALL and METHODRESPONSE. The table below therefore uses the names of these attributes and child elements in the mapping to generic operation parameters.

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		LOCALCLASSPATH child element	N/A	See 1)
MethodName	MethodName	NAME attribute	N/A	
InParmValues	ParameterValue [ ]	set of PARAMVALUE child elements	N/A	

1) The CIM-XML element *LOCALCLASSPATH* includes the model path portion of the class path of the class. The generic parameter *ClassPath* corresponds to the combination of the CIM-XML element *LOCALCLASSPATH* and the target namespace of the CIM-XML operation.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
OutParmValues	ParameterValue [ ]	set of PARAMVALUE child elements	N/A	
ReturnValue	ReturnValue	RETURNVALUE child element	N/A	

Optional behavior: None

Deviations: None

C.1.25 GetClass

CIM-XML Operation Name: GetClass

Purpose: Retrieve a class given its class path.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
N/A	N/A	LocalOnly	boolean	See 2)

1) The CIM-XML parameter *ClassName* specifies the class name. The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of the CIM-XML operation.

2) The defined behavior of generic operation *GetClass* conforms to the behavior of CIM-XML operation *GetClass* with *LocalOnly=false*.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
Class	ClassSpecification-WithPath	return value	class	See 1)

1) The CIM-XML return value includes the class declaration, without any class path information. The generic parameter *Class* needs to contain the class path in addition to the class declaration. A CIM client side mapping layer can remember the class path provided in the generic input parameter *ClassPath*, and add that to the generic output parameter *Class*.

Optional behavior: None

Deviations: None

C.1.26 DeleteClass

CIM-XML Operation Name: DeleteClass

Purpose: Delete a class given its class path.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)
DeleteDependents	boolean	N/A	N/A	See 2)

1) The CIM-XML parameter *ClassName* specifies the class name. The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of the CIM-XML operation.

2) **EXPERIMENTAL:** The experimental generic parameter *DeleteDependents* indicates whether dependent classes and instances are to be deleted as well. DSP0223 defines the generic parameter *DeleteDependents* as optional. CIM-XML does not support deleting dependent classes and instances.

Operation Output Parameters: None

Deviations: None

C.1.27 ModifyClass

CIM-XML Operation Name: ModifyClass

Purpose: Modify a class given its class path.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ModifiedClass	class	See 1)
ModifiedClass	ClassSpecification	ModifiedClass	class	

1) The CIM-XML parameter *ModifiedClass* includes the name of the class that is being modified, and the modified class declaration. The combination of the class name portion of the CIM-XML parameter *ModifiedClass* and the target namespace of the CIM-XML operation corresponds to the generic parameter *ClassPath*.

Operation Output Parameters: None

Optional behavior: None

Deviations: None

C.1.28 CreateClass

CIM-XML Operation Name: CreateClass

Purpose: Create a class.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
NewClass	ClassSpecification	NewClass	class	

Operation Output Parameters: None

Optional behavior: None

Deviations: None

C.1.29 GetTopClassesWithPath

CIM-XML Operation Name: EnumerateClasses with ClassName being NULL

Purpose: Retrieve the top classes (i.e., classes that have no superclasses) of a given namespace. The retrieved classes include their class paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
IncludeSubclasses	boolean	DeepInheritance	boolean	

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
N/A	N/A	ClassName	className	See 1)
N/A	N/A	LocalOnly	boolean	See 2)

1) The defined behavior of generic operation *GetTopClassesWithPath* conforms to the behavior of CIM-XML operation *EnumerateClasses* with *ClassName=NULL*.

2) The defined behavior of generic operation *GetTopClassesWithPath* conforms to the behavior of CIM-XML operation *EnumerateClasses* with *LocalOnly=false*.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassList	ClassSpecification-WithPath [ ]	return value	class [ ]	See 1)

1) The CIM-XML return value includes the set of class declarations including class names, but without a class path. The generic parameter *ClassList* needs to contain the class path in addition to the class declaration. A CIM client side mapping layer can construct the class paths from the class names and the CIM-XML target namespace.

Optional behavior: None

Deviations: None

C.1.30 GetTopClassPaths

CIM-XML Operation Name: EnumerateClassNames with ClassName being NULL

Purpose: Retrieve the class paths of the top classes (i.e., classes that have no superclasses) of a given namespace.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
IncludeSubclasses	boolean	DeepInheritance	boolean	
N/A	N/A	ClassName	className	See 1)

1) The defined behavior of generic operation *GetTopClassPaths* conforms to the behavior of CIM-XML operation *EnumerateClassNames* with *ClassName=NULL*.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	className [ ]	See 1)

1) The CIM-XML return value includes the set of class names, but without a class path. The generic parameter *ClassPathList* needs to contain the class paths. A CIM client side mapping layer can construct the class paths from the class names and the CIM-XML target namespace.

Optional behavior: None

Deviations: None

C.1.31 GetSubClassesWithPath

CIM-XML Operation Name: EnumerateClasses with ClassName being non-NULL

Purpose: Retrieve the subclasses of a given class. The retrieved classes include their class paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1), 2)
IncludeSubclasses	boolean	DeepInheritance	boolean	
IncludeInherited-Elements	boolean	LocalOnly	boolean	See 3)
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	

1) The CIM-XML parameter *ClassName* specifies the class name. The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of the CIM-XML operation.

2) The defined behavior of generic operation *GetSubClassesWithPath* conforms to the behavior of CIM-XML operation *EnumerateClasses* with *ClassName* being *non-NULL*.

3) The generic parameter *IncludeInheritedElements* corresponds to the negated CIM-XML parameter *LocalOnly*.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassList	ClassSpecification-WithPath [ ]	return value	class [ ]	See 1)

1) The CIM-XML return value includes the set of class declarations including class names, but without a class path. The generic parameter *ClassList* needs to contain the class path in addition to the class declaration. A CIM client side mapping layer can construct the class paths from the class names and the CIM-XML target namespace.

Optional behavior: None

Deviations: None

C.1.32 GetSubClassPaths

CIM-XML Operation Name: EnumerateClassNames with ClassName being non-NULL

Purpose: Retrieve the class paths of the subclasses of a given class.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1), 2)
IncludeSubclasses	boolean	DeepInheritance	boolean	

1) The CIM-XML parameter *ClassName* specifies the class name. The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of the CIM-XML operation.

2) The defined behavior of generic operation *GetSubClassPaths* conforms to the behavior of CIM-XML operation *EnumerateClassNames* with *ClassName* being *non-NULL*.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	className [ ]	See 1)

1) The CIM-XML return value includes the set of class names, but without a class path. The generic parameter *ClassPathList* needs to contain the class paths. A CIM client side mapping layer can construct the class paths from the class names and the CIM-XML target namespace.

Optional behavior: None

Deviations: None

C.1.33 GetAssociatedClassesWithPath

CIM-XML Operation Name: Associators with ObjectName being a class path

Purpose: Retrieve the classes that are associated with a given source class. The retrieved classes include their class paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
RoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	

1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter *ObjectName* and the target namespace of the CIM-XML operation.

The generic operation *GetAssociatedClassesWithPath* corresponds to the CIM-XML operation *Associators* when a class path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *Associators* with an instance path for its *ObjectName* parameter is covered by the generic operation *GetAssociatedInstancesWithPath* (see C.1.7).

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassList	ClassSpecification-WithPath [ ]	return value	objectWithPath [ ]	

Optional behavior: None

Deviations: None

C.1.34 GetAssociatedClassPaths

CIM-XML Operation Name: AssociatorNames with ObjectName being a class path

Purpose: Retrieve the class paths of the classes that are associated with a given source class.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	AssocClass	className	
AssociatedClassName	ClassName	ResultClass	className	
RoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	ResultRole	string	

1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter *ObjectName* and the target namespace of the CIM-XML operation.

The generic operation *GetAssociatedClassPaths* corresponds to the CIM-XML operation *AssociatorNames* when a class path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *AssociatorNames* with an instance path for its *ObjectName* parameter is covered by the generic operation *GetAssociatedInstancePaths* (see C.1.8).

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	objectPath [ ]	

Optional behavior: None

Deviations: None

C.1.35 GetReferencingClassesWithPath

CIM-XML Operation Name: References with ObjectName being a class path

Purpose: Retrieve the association classes that reference a given source class. The retrieved classes include their class paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See 2)
RoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	

1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter *ObjectName* and the target namespace of the CIM-XML operation.

The generic operation *GetReferencingClassesWithPath* corresponds to the CIM-XML operation *References* when a class path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *References* with an instance path for its *ObjectName* parameter is covered by the generic operation *GetReferencingInstancesWithPath* (see C.1.9).

2) The CIM-XML operation *References* does not support a means to filter by class name or role name of the associated classes on the other ends of the associations referencing the source class. The generic operation *GetReferencingClassesWithPat*h does support such filtering through its

parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the CIM-XML operation will result in including association classes that these two parameters could filter out, a mapping layer on the CIM client side can implement the behavior defined by these two generic parameters by eliminating association classes if these filter parameters are used.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	objectWithPath [ ]	

Optional behavior: None

Deviations: None

C.1.36 GetReferencingClassPaths

CIM-XML Operation Name: ReferenceNames with ObjectName being a class path

Purpose: Retrieve the class paths of the association classes that reference a given class.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPath	ClassPath	target namespace	N/A	See 1)
		ObjectName	objectName	See 1)
AssociationClassName	ClassName	ResultClass	className	
AssociatedClassName	ClassName	N/A	N/A	See 2)
RoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)

1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter *ObjectName* and the target namespace of the CIM-XML operation.

   The generic operation *GetReferencingClassPaths* corresponds to the CIM-XML operation *ReferenceNames* when a class path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *ReferenceNames* with an instance path for its *ObjectName* parameter is covered by the generic operation *GetReferencingInstancePaths* (see C.1.10).

2) The CIM-XML operation *References* does not support a means to filter by class name or role name of the associated classes on the other ends of the associations referencing the source class. The generic operation *GetReferencingClassesWithPath* does support such filtering through its parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the CIM-XML operation will result in including association classes that these two parameters could filter out, a mapping layer on the CIM client side can implement the behavior defined by these two generic parameters by eliminating association classes if these filter parameters are used.

5615 **Operation Output Parameters:**

5616

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	objectPath [ ]	

5617 **Optional behavior:** None

5618 **Deviations:** None

5619 ### C.1.37 GetQualifierType

5620 **CIM-XML Operation Name:** GetQualifier

5621 **Purpose:** Retrieve a qualifier type given its qualifier type path.

5622 **Operation Input Parameters:**

5623

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypePath	QualifierTypePath	target namespace	N/A	See 1), 1)
		QualifierName	string	See 1), 1)

5624 1) The CIM-XML parameter *QualifierName* specifies the name of the qualifier type. The generic
5625 parameter *QualifierTypePath* corresponds to the combination of the CIM-XML parameter
5626 *QualifierName* and the target namespace of the CIM-XML operation.

5627 **Operation Output Parameters:**

5628

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierType	QualifierType	return value	qualifierDecl	See 1)

5629 1) The CIM-XML return value includes the qualifier type declaration including the qualifier type name,
5630 but without the namespace path portion of the full qualifier type path. The generic parameter
5631 *QualifierType* needs to contain the full qualifier type path in addition to the qualifier type declaration.
5632 A CIM client side mapping layer can remember the qualifier type path provided in the generic input
5633 parameter *QualifierTypePath*, and add that to the generic output parameter *QualifierType*.

5634 **Optional behavior:** None

5635 **Deviations:** None

5636 ### C.1.38 DeleteQualifierType

5637 **CIM-XML Operation Name:** DeleteQualifier

5638 **Purpose:** Delete a qualifier type given its qualifier type path.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypePath	QualifierTypePath	target namespace	N/A	See 1)
		QualifierName	string	See 1)

1) The CIM-XML parameter *QualifierName* specifies the name of the qualifier type, i.e. the model path portion of its qualifier type path. The generic parameter *QualifierTypePath* corresponds to the combination of the CIM-XML parameter *QualifierName* and the target namespace of the CIM-XML operation.

Operation Output Parameters: None

Deviations: None

C.1.39 ModifyQualifierType

CIM-XML Operation Name: SetQualifier

Purpose: Modify a qualifier type given its qualifier type path.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypePath	QualifierTypePath	target namespace	N/A	See 1)
		QualifierDeclaration	qualifierDecl	See 1)
ModifiedQualifierType	QualifierType	QualifierDeclaration	qualifierDecl	

1) The CIM-XML parameter *QualifierDeclaration* includes the name of the qualifier type that is modified, i.e. the model path portion of its qualifier type, and the modified qualifier type declaration. The combination of the name of the qualifier type within the CIM-XML parameter *QualifierDeclaration* and the target namespace of the CIM-XML operation corresponds to the generic parameter *QualifierTypePath*.

Operation Output Parameters: None

Optional behavior: None

Deviations:

- The generic operation *ModifyQualifierType* is required to fail if invoked on a non-existing qualifier type. The CIM-XML operation *SetQualifier* creates the qualifier type in this case. This deviation covers only an error case. A CIM client side mapping layer can expose the generic operation behavior by first testing for the existence of the qualifier type using the CIM-XML operation *GetQualifier*, before modifying it.

C.1.40 CreateQualifierType

CIM-XML Operation Name: SetQualifier

Purpose: Create a CIM qualifier type.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	See 1)
		QualifierDeclaration	qualifierDecl	See 1)
NewQualifierType	QualifierType	QualifierDeclaration	qualifierDecl	

1) The generic parameter *NamespacePath* corresponds to the combination of the qualifier type name specified in the CIM-XML parameter *NewQualifierType* and the target namespace of the CIM-XML operation.

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypePath	QualifierTypePath	return value	instanceName	

Optional behavior: None

Deviations:

- The generic operation *CreateQualifierType* is required to fail if invoked on an existing qualifier type. The CIM-XML operation *SetQualifier* modifies the qualifier type in this case. This deviation covers only an error case. A CIM client side mapping layer can expose the generic operation behavior by first testing for the existence of the qualifier type using the CIM-XML operation *GetQualifier*, before creating it.

C.1.41 EnumerateQualifierTypesWithPath

CIM-XML Operation Name: EnumerateQualifiers

Purpose: Retrieve the qualifier types of a given namespace. The retrieved qualifier types include their qualifier type paths.

Operation Input Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	

Operation Output Parameters:

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypeList	QualifierTypeWithPath [ ]	return value	qualifierDecl [ ]	See 1)

1) The CIM-XML return value includes the set of qualifier type declarations including their names, but without namespace paths. The generic parameter QualifierTypeList needs to contain the qualifier type paths in addition to the set of qualifier type declarations. A CIM client side mapping layer can construct the qualifier type paths from the qualifier names and the CIM-XML target namespace.

Optional behavior: None

Deviations: None

ANNEX D
(informative)

Change Log

Version	Date	Description
1.0	1999-06-02	DMTF Final Standard
1.1	2003-01-06	DMTF Final Standard
1.2	2007-01-09	DMTF Final Standard
1.3.0	2008-10-15	DMTF Final Standard
1.3.1	2009-07-29	DMTF Standard Release

Version	Date	Description
1.4.0a	2013-05-07	Work in Progress, with the following changes: Changes: • Changed representation of enumeration context value from an ENUMERATIONCONTEXT element to a string using the VALUE element (see 5.4.2.24.2) *(CRCIMXML00022.001)* • Added requirement to support DMTF Filter Query Language (FQL) in pulled enumeration operations (see 5.4.2.24.2) *(CRCIMXML00033.001)* • Updated several normative references (see clause 2) *(multiple CRs)* • Lifted requirements in CreateInstance to initialize only with client-provided values, and in ModifyInstance to update only with client-provided values, to leave room for model-defined deviations (see 5.4.2.6 and 5.4.2.8). *(CRCIMXML00036.000)* Deprecations:: • Deprecated use of CIM_ERR_INVALID_CLASS on ExportIndication operation (see 5.5.2.1) *(CRCIMXML00021.000)* • Deprecated the GetProperty and SetProperty operations (see 5.4.2.18 and 5.4.2.19) *(CRCIMXML00027.000)* • Deprecated the EnumerateInstances, EnumerateInstanceNames, ExecQuery, and the instance-level Associators, AssociatorNames, References and ReferenceNames operations *(CRCIMXML00030.002)* Additional Functions and Requirements: • Added support for operation correlators (see 5.3) *(CRCIMXML00014.002)* Clarifications: • Clarified HTTPS support (see 7.1) *(CRCIMXML00010.004)* • Clarified filter query in pulled enumerations (5.4.2.24.2) *(CRCIMXML00019.001)* • Added mapping to generic operations (see ANNEX C) *(CRCIMXML00034.000)* Editorial Changes: • Terminology cleanup *(CRCIMXML00026.002)*

5701

Bibliography

DMTF DSP0203, *DTD for Representation of CIM in XML 2.4*, http://www.dmtf.org/standards/published_documents/DSP0203_2.4.dtd

DMTF DSP8044, *XSD for Representation of CIM in XML 2.4*, http://schemas.dmtf.org/wbem/wbem/cim-xml/2/dsp8044_2.4.xsd

IETF RFC2068 (obsoleted by RFC2616), *Hypertext Transfer Protocol – HTTP/1.1*, January 1997, http://www.ietf.org/rfc/rfc2068.txt

IETF RFC2069 (obsoleted by RFC2617), *An Extension to HTTP: Digest Access Authentication*, January 1997, http://www.ietf.org/rfc/rfc2069.txt

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

SSL 2.0, *Hickman: The SSL Protocol, Draft 02, Netscape Communications Corp.*, February 1995, http://www.mozilla.org/projects/security/pki/nss/ssl/draft02.html

SSL 3.0, *Freier, Karlton, and Kocher: The SSL Protocol, Version 3.0, Final Draft, Netscape Communications Corp.*, November 1996, http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt