1

CIM Operations over HTTP

10   Copyright Notice

11   Copyright © 1999-2013 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

32

CONTENTS

214 **Tables**

225   Foreword

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

227 # Introduction

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

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

233 This document is structured as follows:

234 • Clause 5 describes the CIM-XML messages that form the HTTP payload using XML. It specifies
235 the syntax and semantics of the message requests and their corresponding responses.

236 • Clause 6 describes the encapsulation of these messages in HTTP request and response
237 messages, with examples of each. It also describes the extension headers used to convey
238 additional CIM-specific semantics in the HTTP Header.

239 • Clause 7 presents details of other aspects of the encapsulation:

240 – HTTP version support

241 – Use of standard HTTP headers

242 – HTTP error codes

243 – Security considerations

244 ## Requirements

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

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

251 • Each CIM-XML message is completely described in XML; completeness is favored over
252 conciseness.

253 • The set of CIM-XML messages provides enough functionality to enable implementations of CIM
254 to communicate effectively for management purposes. This release of the mapping does not
255 provide a *complete* set of messages. Rather, the goal is to define the mapping so that it admits
256 straightforward extension (by the addition of further features) in future versions.

257 • **(DEPRECATED)** The set of CIM-XML messages is classified into functional profiles to
258 accommodate a range of implementations varying from complete support of all messages to
259 support of a minimal subset. The number of functional profiles is kept as small as possible to
260 encourage interoperability, and mechanisms provided by different CIM implementations can
261 declare their level of support.

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

263 • In recognition of the large installed base of HTTP/1.0 systems, the encapsulation is designed to
264 support both HTTP/1.0 and HTTP/1.1. However, support for HTTP/1.0 has been deprecated in
265 version 1.4 of this document (see 7.1).

266 • The encapsulation does not introduce requirements that conflict with those stated in HTTP/1.0
267 or HTTP/1.1.

268 • Use of the encapsulation should be straightforward over the current base HTTP infrastructures.
269   Some features anticipate and exploit enhancements to this base, but no aspects of the
270   encapsulation require such enhancements as mandatory.

271 • The encapsulation avoids the use of pure HTTP tunneling or URL munging (for example, the
272   use of the "?" character) in favor of a mechanism that allows existing HTTP infrastructures to
273   control content safely.

274 • The encapsulation exposes key CIM-XML message information in headers to allow efficient
275   firewall/proxy handling. The information is limited to essentials so that it does not have a
276   significant impact on the size of the header. All CIM-specific information in a header also
277   appears within the CIM-XML message.

278 • There is a clear and unambiguous encapsulation of the CIM-XML message payload within the
279   HTTP message. Conciseness of the encapsulation is of secondary importance.
280

281

282 # CIM Operations over HTTP

283 ## 1 Scope

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

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

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

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

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

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

300 ## 2 Normative References

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

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

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

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

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

312 DMTF DSP8016, *WBEM Operations Message Registry 1.0*,
313 http://schemas.dmtf.org/wbem/messageregistry/1/dsp8016_1.0.xml

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

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

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

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

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

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

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

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

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

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

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

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

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

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

342 NIST 800-131A, *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and*
343 *Key Lengths*, January 2011,
344 http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

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

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

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

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

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

355 # 3   Terms and Definitions

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

358 The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"),
359 "may", "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described

360   in ISO/IEC Directives, Part 2, Annex H. The terms in parenthesis are alternatives for the preceding term,
361   for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note
362   thatISO/IEC Directives, Part 2, Annex H specifies additional alternatives. Occurrences of such additional
363   alternatives shall be interpreted in their normal English meaning.

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

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

369   The terms defined in DSP0004 and DSP0201 apply to this document. The following additional terms are
370   used in this document. Some additional more detailed terms are defined throughout the subclauses of
371   this document.

372   **3.1**
373   **CIM element**
374   one of the following components of the CIM metamodel used to define a schema: Class, instance,
375   property, method, parameter, or qualifier

376   **3.2**
377   **CIM object**
378   a namespace, class, instance, or qualifier that is accessible in a WBEM server

379
380   **CIM-XML protocol**
381   the WBEM protocol that uses the CIM operations over HTTP defined in this document and the
382   representation of CIM in XML defined in DSP0201

383   **3.3**
384   **WBEM client**
385   the client role in the CIM-XML protocol and in other WBEM protocols. See 6.1 for a complete definition.

386   **3.4**
387   **WBEM listener**
388   the event listener role in the CIM-XML protocol and in other WBEM protocols. See 6.1 for a complete
389   definition.

390   **3.5**
391   **WBEM protocol**
392   a communications protocol between WBEM client, WBEM server and WBEM listener

393   **3.6**
394   **WBEM server**
395   the server role in the CIM-XML protocol and in other WBEM protocols. See 6.1 for a complete definition.

396   **3.7**
397   **XML element**
398   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.

4.1.1

CIM

Common Information Model

4.1.2

DTD

Document Type Definition

4.1.3

HTTP

Hypertext Transfer Protocol

4.1.4

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).

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

5.1   Well-Formed, Valid, and Loosely Valid Documents

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

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

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

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

457   Instead, the following requirements are set forth in this document. In all cases, a WBEM client has a
458   respective alternative WBEM server or WBEM listener, and a received CIM-XML message response has
459   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.

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

5.2   Operational Semantics

469   The CIM Representation in XML (DSP0201) defines a child element under the root <CIM> XML element
470   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>

478   –   \<SIMPLEXPRSP\>

479   –   \<MULTIEXPREQ\>

480   –   \<MULTIEXPRSP\>

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

483   •   *Operation request message*. Contains under the root \<CIM\> node a \<MESSAGE\> child
484   element that has a \<MULTIREQ\> or \<SIMPLEREQ\> child element under it.

485   •   *Operation response message*. Contains under the root \<CIM\> node a \<MESSAGE\> child
486   element that has a \<MULTIRSP\> or \<SIMPLERSP\> child element under it.

487   •   *Export request message*. Contains under the root \<CIM\> node a \<MESSAGE\> child element
488   that has a \<MULTIEXPREQ\> or \<SIMPLEEXPREQ\> child element under it.

489   •   *Export response message*. Contains under the root \<CIM\> node a \<MESSAGE\> child element
490   that has a \<MULTIEXPRSP\> or \<SIMPLEEXPRSP\> child element under it.

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

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

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

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

504   –   A \<MULTIRSP\> child element, if the operation request message contains a \<MULTIREQ\>
505   child element.

506   –   A \<SIMPLERSP\> child element, if the operation request message contains a
507   \<SIMPLEREQ\> child element.

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

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

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

517   –   A \<MULTIEXPRSP\> child element if the export request message contained a
518   \<MULTIEXPREQ\> child element, or

519   –   A \<SIMPLEEXPRSP\> child element if the export request message contained a
520   \<SIMPLEEXPREQ\> child element.

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

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

527 5.3   Operation Correlators

528 5.3.1   Overview

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

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

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

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

547 5.3.2   Representation

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

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

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

555 5.3.3   Implementation Requirements and Compatibility for Operation Messages

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

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

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

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

5.3.4 Implementation Requirements and Compatibility for Export Messages

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

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

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

5.4 CIM Operation Syntax and Semantics

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

5.4.1 Method Invocations

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

587 • An *intrinsic* method, which is defined for the purposes of modeling a CIM operation.

588 • An *extrinsic* method, which is defined on a CIM class in a schema.

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

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

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

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

599 • Each input parameter shall be named using the name assigned in the method definition.

600 • Input parameters may be supplied in any order.

601 • Each input parameter of the method, and no others, shall be present in the call, unless it is
602 optional.

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

606 • Each output parameter shall be named using the name assigned in the method definition.

607 • Output parameters may be supplied in any order.

608 • Each output parameter of the method, and no others, shall be present in the response, unless it
609 is optional.

610 • The method invocation process can be thought of as the binding of the input parameter values
611 specified as child elements of the <METHODCALL> or <IMETHODCALL> element to the input
612 parameters of the method. This binding is followed by an attempt to execute the method using
613 the bound input parameters with one of the following results:

614 – If the attempt to call the method is successful, the return value and output parameters are
615 bound to the child elements of the <METHODRESPONSE> or <IMETHODRESPONSE>
616 element.

617 – If the attempt to call the method is unsuccessful, an error code and optional human-
618 readable description of that code is bound to the <METHODRESPONSE> or
619 <IMETHODRESPONSE> element.

620 5.4.1.1 Simple Operations

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

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

627 • A <LOCALCLASSPATH> child element identifying the CIM class on which the method is to be
628 invoked if the method is static.

629 • A <LOCALINSTANCEPATH> child element identifying the CIM instance on which the method is
630 otherwise to be invoked.

631 5.4.1.2 Multiple Operations

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

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

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

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

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

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

651 **5.4.1.3 Status Codes**

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

655   •   a mandatory status code

656   •   an optional human-readable description of the status code

657   •   zero or more CIM_Error instances

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

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

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

669   **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.

670   5.4.2   Intrinsic Methods

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

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

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

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

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

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

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

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

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

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

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

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

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

702   **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

703 **5.4.2.1   GetClass**

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

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

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

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

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

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

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

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

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

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

739   • CIM_ERR_ACCESS_DENIED

740   • CIM_ERR_INVALID_NAMESPACE

741   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise
742   incorrect parameters)

743   • CIM_ERR_NOT_FOUND (The request CIM class does not exist in the specified namespace.)

744   • CIM_ERR_FAILED (Some other unspecified error occurred.)

745 **5.4.2.2   GetInstance**

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

```
747     <instance> GetInstance (
748         [IN] <instanceName> InstanceName,
749         [IN,OPTIONAL] boolean LocalOnly = true,          (DEPRECATED)
750         [IN,OPTIONAL] boolean IncludeQualifiers = false,  (DEPRECATED)
751         [IN,OPTIONAL] boolean IncludeClassOrigin = false, (DEPRECATED)
752         [IN,OPTIONAL,NULL] string PropertyList [] = NULL
753     )
```

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

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

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

775 DEPRECATION NOTE: In version 1.4 of this document, the `IncludeClassOrigin` parameter is
776 DEPRECATED. A WBEM server may choose to treat the value of `IncludeClassOrigin` parameter as
777 false for all requests, otherwise the implementation shall support the original behavior as defined in the
778 rest of this paragraph.If the `IncludeClassOrigin` input parameter is `true`, the CLASSORIGIN attribute
779 shall be present on all appropriate elements in the returned instance. If it is `false`, no CLASSORIGIN
780 attributes are present.

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

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

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

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

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

801 • CIM_ERR_ACCESS_DENIED

802   • CIM_ERR_INVALID_NAMESPACE

803   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
804   incorrect parameters)

805   • CIM_ERR_INVALID_CLASS (The CIM class does not exist in the specified namespace.)

806   • CIM_ERR_NOT_FOUND (The CIM class does exist, but the requested CIM instance does not
807   exist in the specified namespace.)

808   • CIM_ERR_FAILED (some other unspecified error occurred)

809   **5.4.2.3   DeleteClass**

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

811   void **DeleteClass** (
812   [IN] <className> ClassName
813   )

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

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

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

820   • CIM_ERR_ACCESS_DENIED

821   • CIM_ERR_NOT_SUPPORTED

822   • CIM_ERR_INVALID_NAMESPACE

823   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
824   incorrect parameters)

825   • CIM_ERR_NOT_FOUND (The CIM class to be deleted does not exist.)

826   • CIM_ERR_CLASS_HAS_CHILDREN (The CIM class has one or more subclasses that cannot
827   be deleted.)

828   • CIM_ERR_CLASS_HAS_INSTANCES (The CIM class has one or more instances that cannot
829   be deleted.)

830   • CIM_ERR_FAILED (Some other unspecified error occurred.)

831   **5.4.2.4   DeleteInstance**

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

833   void **DeleteInstance** (
834   [IN] <instanceName> InstanceName
835   )

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

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

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

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

845   • CIM_ERR_ACCESS_DENIED

846   • CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)

847   • CIM_ERR_INVALID_NAMESPACE

848   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
849   incorrect parameters)

850   • CIM_ERR_INVALID_CLASS (The CIM class does not exist in the specified namespace.)

851   • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
852   instance, if provided.)

853   • CIM_ERR_NOT_FOUND (The CIM class does exist, but the requested CIM instance does not
854   exist in the specified namespace.)

855   • CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
856   unspecified error occurred.)

857   **5.4.2.5   CreateClass**

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

```
860       void CreateClass (
861          [IN] <class> NewClass
862       )
```

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

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

866   • The server shall ignore any CLASSORIGIN and PROPAGATED XML attributes in the new
867   class.

868   • If the new class has no superclass, the `NewClass` parameter defines a new superclass. The
869   server shall ensure that all properties and methods of the new class have a CLASSORIGIN
870   attribute whose value is the name of the new class.

871   • If the new class has a superclass, the `NewClass` parameter defines a new subclass of that
872   superclass. The superclass shall exist. The server shall ensure that the following conditions are
873   met:

874   – Any properties, methods, or qualifiers in the subclass not defined in the superclass are
875   created as new elements of the subclass. In particular, the server shall set the
876   CLASSORIGIN XML attribute on the new properties and methods to the name of the
877   subclass and ensure that all others preserve their CLASSORIGIN attribute value from that
878   defined in the superclass.

879   – If a property is defined in the superclass and in the subclass, the value assigned to that
880   property in the subclass (including NULL) becomes the default value of the property for the
881   subclass.

882   – If a property or method of the superclass is not specified in the subclass, then it is inherited
883   without modification by the subclass.

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

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

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

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

896 • CIM_ERR_ACCESS_DENIED

897 • CIM_ERR_NOT_SUPPORTED

898 • CIM_ERR_INVALID_NAMESPACE

899 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
900 incorrect parameters)

901 • CIM_ERR_ALREADY_EXISTS (The CIM class already exists.)

902 • CIM_ERR_INVALID_SUPERCLASS (The putative CIM class declares a non-existent
903 superclass.)

904 • CIM_ERR_FAILED (Some other unspecified error occurred.)

905 **5.4.2.6   CreateInstance**

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

908 <instanceName> **CreateInstance** (
909   [IN] <instance> NewInstance
910)

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

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

919 • The server shall ignore any CLASSORIGIN and PROPAGATED XML attributes in the
920 NewInstance.

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

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

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

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

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

935   •   If the `NewInstance` parameter specifies properties that are not exposed by the class specified
936   in the `NewInstance` parameter, the server shall reject the request.

937   •   **DEPRECATION NOTE:** Use of the TOINSTANCE attribute is DEPRECATED. Servers may
938   choose to ignore TOINSTANCE. Servers that do not ignore TOINSTANCE shall interpret it so
939   that any qualifiers defined in the class with a TOINSTANCE attribute value of `true` appear in
940   the instance. Qualifiers in the class with a value of `false` shall not be propagated to the
941   instance.

942   •   **DEPRECATED.** Any Qualifier propagated from the class cannot be modified in the instance if
943   the OVERRIDABLE attribute of that qualifier is set to `false` in the class. It is a client error to
944   specify such a qualifier in the `NewInstance` with a definition different than that in the class
945   (where definition encompasses the name, type, and flavor attribute settings of the
946   <QUALIFIER> XML element and the value of the qualifier).

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

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

955   •   CIM_ERR_ACCESS_DENIED

956   •   CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)

957   •   CIM_ERR_INVALID_NAMESPACE

958   •   CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
959   incorrect parameters)

960   •   CIM_ERR_INVALID_CLASS (The CIM class for the new instance does not exist.)

961   •   CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
962   instance, if provided.)

963   •   CIM_ERR_ALREADY_EXISTS (The CIM instance already exists.)

964   •   CIM_ERR_FAILED (This operation is not supported for the specified instance or some other
965   unspecified error occurred.)

966   **5.4.2.7**   **ModifyClass**

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

969   void **ModifyClass** (
970   [IN] <class> ModifiedClass

971   )

972   The `ModifiedClass` input parameter defines the set of changes to be made to the current class
973   definition, which shall be correct amendments to the CIM class as defined by [DSP0004](#).

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

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

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

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

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

- If the modified class has a superclass, the `ModifiedClass` parameter defines modifications to
  a subclass of that superclass. The superclass shall exist, and the client shall not change the
  name of the superclass in the modified subclass. 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 elements of the subclass. In particular, the server shall set the CLASSORIGIN
    attribute on the new properties and methods to the name of the subclass and shall ensure
    that all other others preserve their CLASSORIGIN attribute value from that defined in the
    superclass.

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

  – If a property is specified in the `ModifiedClass` parameter, the value assigned to that
    property (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 the subclass
    inherits it without modification. Any previous changes to such an element in the subclass
    are lost.

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

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

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

1011   If ModifyClass is successful, the WBEM server updates the specified class. The request to modify the
1012   class shall fail if the server cannot consistently update any existing subclasses or instances of that class.

1013   If ModifyClass is unsuccessful, this method shall return one of the following status codes, where the error
1014   returned is the first applicable error in the list, starting with the first element and working down. Any
1015   additional method-specific interpretation of the error is enclosed in parentheses.

1016 • CIM_ERR_ACCESS_DENIED

1017 • CIM_ERR_NOT_SUPPORTED

1018 • CIM_ERR_INVALID_NAMESPACE

1019 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1020   incorrect parameters)

1021 • CIM_ERR_NOT_FOUND (The CIM class does not exist.)

1022 • CIM_ERR_INVALID_SUPERCLASS (The putative CIM class declares a non-existent or
1023   incorrect superclass.)

1024 • CIM_ERR_CLASS_HAS_CHILDREN (The modification could not be performed because the
1025   subclasses of the class could not be updated consistently.)

1026 • CIM_ERR_CLASS_HAS_INSTANCES (The modification could not be performed because the
1027   instances of the class could not be updated consistently.)

1028 • CIM_ERR_FAILED (Some other unspecified error occurred.)

1029 **5.4.2.8 ModifyInstance**

1030 The ModifyInstance operation modifies an existing CIM instance in the target namespace. The instance
1031 shall already exist:

```
1032    void ModifyInstance (
1033        [IN] <namedInstance> ModifiedInstance,
1034        [IN, OPTIONAL] boolean IncludeQualifiers = true,     (DEPRECATED)
1035        [IN, OPTIONAL, NULL] string PropertyList[] = NULL
1036    )
```

1037 The ModifiedInstance input parameter identifies the name of the instance to be modified and
1038 provides the new property values.

1039 **DEPRECATION NOTE:** Use of the IncludeQualifiers parameter is DEPRECATED, and it may be
1040 removed in a future version of this document. The behavior of the IncludeQualifiers parameter is
1041 not specified. A WBEM client cannot rely on IncludeQualifiers to have any impact on the operation.
1042 It is recommended that the WBEM server ignore any qualifiers included in ModifiedInstance. If the
1043 IncludeQualifiers input parameter is true, the qualifiers are modified as specified in
1044 ModifiedInstance. If the parameter is false, qualifiers in ModifiedInstance are ignored and no
1045 qualifiers are explicitly modified.

1046 The set of properties designated to be modified shall be determined as follows:

1047 If the PropertyList input parameter is not NULL, the members of the array define one or more
1048 property names. The properties specified in PropertyList are designated to be modified. Properties of
1049 the ModifiedInstance that are missing from PropertyList are not designated to be modified. If
1050 PropertyList is an empty array, no properties are designated to be modified. If PropertyList is
1051 NULL, the properties of ModifiedInstance with values different from the current values in the instance
1052 are designated to be modified.

1053 If PropertyList contains duplicate property names, the WBEM server shall ignore them but otherwise
1054 process the request normally. If PropertyList contains property names that are invalid for the instance
1055 to be modified, the WBEM server shall reject the request.

1056 If a property is designated to be modified, the WBEM server shall either modify the property, or reject the
1057 request. The server shall reject modification requests for key properties. Further considerations for
1058 accepting or rejecting modification requests based on the properties requested to be modified are out of

1059   scope for this document; CIM model definitions are expected to cover that. Note that the WRITE qualifier
1060   on a property is considered to be in the area of CIM models; specifically, a value of True for the WRITE
1061   qualifier does not guarantee modifiability of that property, and a value of False does not prevent
1062   modifiability.

1063   If a property is not designated to be modified, the server shall not modify its value. However, note that
1064   properties may change their values as a result of other changes.

1065   In modifying the instance, the WBEM server shall conform to the following rules and ensure their
1066   application:

1067   •   The server shall ignore any CLASSORIGIN and PROPAGATED attributes in the
1068   ModifiedInstance.

1069   •   The class shall exist, and the client shall not change its name in the instance to be modified.

1070   •   **DEPRECATED.** Any qualifiers in the instance not defined in the class are created as new
1071   elements of the instance if `IncludeQualifiers` is `true`.

1072   •   All properties of the instance to be modified preserve their CLASSORIGIN attribute value from
1073   that defined in the class.

1074   •   **DEPRECATED.** Any qualifier previously defined in the instance to be modified but not defined
1075   in the class, and which is not present in the `ModifiedInstance` parameter, is removed from
1076   the instance if `IncludeQualifiers` is `true`.

1077   •   If a property is to be modified as previously defined, the designated new value for that property
1078   in the CIM instance shall be the property value (including NULL) specified in the
1079   `ModifiedInstance` parameter, or if the property is not specified in the `ModifiedInstance`
1080   parameter, the default value (including NULL) defined in the property declaration, or if the
1081   property does not define a default value, there is no designated new value for the property.

1082   If there is a designated new value for a property, the server shall either update the property to
1083   that value, or reject the request. If there is no designated new value for a property, the server
1084   may update the property to any value (including NULL). Further determinations about this
1085   decision are out of scope for this document; CIM model definitions are expected to cover that..

1086   •   **DEPRECATION NOTE:** The use of the TOINSTANCE qualifier attribute is DEPRECATED.
1087   Servers may choose to ignore TOINSTANCE. Servers that do not ignore TOINSTANCE shall
1088   interpret it so that any qualifiers defined in the class with a TOINSTANCE attribute value of true
1089   appear in the instance. A propagated qualifier cannot be removed from an instance. qualifiers in
1090   the class with a TOINSTANCE attribute value of `false` shall not be propagated to the instance

1091   •   **DEPRECATED.** Any qualifier propagated from the class cannot be modified in the instance if
1092   the OVERRIDABLE attribute of that qualifier is set to false in the class. It is a client error to
1093   specify such a qualifier in `ModifiedInstance` with a definition different than that in the class
1094   (where definition encompasses the name, type, and flavor attribute settings of the
1095   <QUALIFIER> XML element and the value of the qualifier).

1096   If ModifyInstance is successful, the specified CIM instance has been updated as described in this
1097   subclause.

1098   If ModifyInstance is unsuccessful, the specified Instance is not updated, and the method shall return one
1099   of the following status codes, where the error returned is the first applicable error in the list, starting with
1100   the first element and working down. Any additional interpretation of the error is enclosed in parentheses.

1101   •   CIM_ERR_ACCESS_DENIED

1102   •   CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)

1103   •   CIM_ERR_INVALID_NAMESPACE

1104 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1105   incorrect parameters and invalid properties to be modified)

1106 • CIM_ERR_INVALID_CLASS (The CIM class of the instance to be modified does not exist.)

1107 • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1108   instance, if provided.)

1109 • CIM_ERR_NOT_FOUND (The CIM instance to be modified does not exist.)

1110 • CIM_ERR_FAILED (This operation is not supported for the specified instance or some other
1111   unspecified error occurred, including a request for non-writable properties to be modified or a
1112   property that cannot be modified at this time.)

1113 5.4.2.9 EnumerateClasses

1114 The EnumerateClasses operation enumerates subclasses of a CIM class in the target namespace:

```
1115    <class>* EnumerateClasses (
1116        [IN,OPTIONAL,NULL] <className> ClassName=NULL,
1117        [IN,OPTIONAL] boolean DeepInheritance = false,
1118        [IN,OPTIONAL] boolean LocalOnly = true,
1119        [IN,OPTIONAL] boolean IncludeQualifiers = true,
1120        [IN,OPTIONAL] boolean IncludeClassOrigin = false
1121    )
```

1122 The `ClassName` input parameter defines the class that is the basis for the enumeration.

1123 If the `DeepInheritance` input parameter is `true`, all subclasses of the specified class should be
1124 returned. If the `ClassName` input parameter is absent, this implies that all classes in the target
1125 namespace should be returned. If `DeepInheritance` is `false`, only immediate child subclasses are
1126 returned. If the `ClassName` input parameter is NULL, this implies that all top-level classes (that is,
1127 classes with no superclass) in the target namespace should be returned. This definition of
1128 `DeepInheritance` applies only to the EnumerateClasses and EnumerateClassName operations.

1129 If the `LocalOnly` input parameter is `true`, any CIM elements (properties, methods, and qualifiers)
1130 except those added or overridden in the class as specified in the classname input parameter shall not be
1131 included in the returned class. If it is `false`, this parameter defines no additional filtering.

1132 If the `IncludeQualifiers` input parameter is `true`, all qualifiers for each class (including qualifiers on
1133 the class and on any returned properties, methods, or method parameters) shall be included as
1134 <QUALIFIER> XML elements in the response. If it is `false`, no <QUALIFIER> XML elements are
1135 present.

1136 If the `IncludeClassOrigin` input parameter is `true`, the CLASSORIGIN attribute shall be present on
1137 all appropriate elements in each returned class. If it is `false`, no CLASSORIGIN attributes are present.

1138 If EnumerateClasses is successful, the method returns zero or more classes that meet the required
1139 criteria. These classes shall include all CIM elements (properties, methods, and qualifiers) defined in or
1140 inherited by each class, reduced by any elements excluded as a result of using the `LocalOnly` filter.

1141 If EnumerateClasses is unsuccessful, this method shall return one of the following status codes, where
1142 the error returned is the first applicable error in the list, starting with the first element and working down.
1143 Any additional method-specific interpretation of the error is enclosed in parentheses.

1144 • CIM_ERR_ACCESS_DENIED

1145 • CIM_ERR_NOT_SUPPORTED

1146   • CIM_ERR_INVALID_NAMESPACE

1147   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1148   incorrect parameters)

1149   • CIM_ERR_INVALID_CLASS (The CIM class for this enumeration does not exist.)

1150   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1151   **5.4.2.10 EnumerateClassNames**

1152   The EnumerateClassNames operation enumerates the names of subclasses of a CIM class in the target
1153   namespace:

1154   <className>* **EnumerateClassNames** (
1155   [IN,OPTIONAL,NULL] <className> ClassName = NULL,
1156   [IN,OPTIONAL] boolean DeepInheritance = false
1157   )

1158   The ClassName input parameter defines the class that is the basis for the enumeration.

1159   If the DeepInheritance input parameter is true, the names of all subclasses of the specified class
1160   should be returned. If the ClassName input parameter is absent, this implies that the names of all classes
1161   in the target namespace should be returned. If DeepInheritance is false, only the names of immediate
1162   child subclasses are returned. If the ClassName input parameter is NULL, this implies that the names of
1163   all top-level classes (that is, classes with no superclass) in the target namespace should be returned. This
1164   definition of DeepInheritance applies only to the EnumerateClasses and EnumerateClassName
1165   operations.

1166   If EnumerateClassNames is successful, the method returns zero or more names of classes that meet the
1167   requested criteria.

1168   If EnumerateClassNames is unsuccessful, this method returns one of the following status codes, where
1169   the error returned is the first applicable error in the list, starting with the first element and working down.
1170   Any additional method-specific interpretation of the error is enclosed in parentheses.

1171   • CIM_ERR_ACCESS_DENIED

1172   • CIM_ERR_NOT_SUPPORTED

1173   • CIM_ERR_INVALID_NAMESPACE

1174   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1175   incorrect parameters)

1176   • CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not
1177   exist.)

1178   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1179   **5.4.2.11 EnumerateInstances (DEPRECATED)**

1180   The EnumerateInstances operation enumerates instances of a CIM class in the target namespace,
1181   including instances in the class and any subclasses in accordance with the polymorphic nature of CIM
1182   objects:

1183   <namedInstance>* **EnumerateInstances** (
1184   [IN] <className> ClassName,
1185   [IN,OPTIONAL] boolean LocalOnly = true,   **(DEPRECATED)**
1186   [IN,OPTIONAL] boolean DeepInheritance = true,
1187   [IN,OPTIONAL] boolean IncludeQualifiers = false, **(DEPRECATED)**

```
1188          [IN,OPTIONAL] boolean IncludeClassOrigin = false, (DEPRECATED)
1189          [IN,OPTIONAL,NULL] string PropertyList [] = NULL
1190      )
```

1191   **DEPRECATION NOTE:** The EnumerateInstances operation has been deprecated in version 1.4 of this
1192   document. Use OpenEnumerateInstances instead (see 5.4.2.24.3).

1193   The `ClassName` input parameter defines the class that is the basis for the enumeration.

1194   **DEPRECATION NOTE:** With version 1.2of this document, the `LocalOnly` parameter is DEPRECATED.
1195   `LocalOnly` filtering, as defined in 1.1, will not be supported in the next major revision of this document.
1196   In version 1.1of this document, the definition of the `LocalOnly` parameter was incorrectly modified. This
1197   change introduced a number of interoperability and backward compatibility problems for WBEM clients
1198   using the `LocalOnly` parameter to filter the set of properties returned. The DMTF strongly recommends
1199   that WBEM clients set `LocalOnly` to `false` and do not use this parameter to filter the set of properties
1200   returned. To minimize the impact of this recommendation on WBEM clients, a WBEM server may choose
1201   to treat the value of the `LocalOnly` parameter as `false` for all requests. A WBEM server shall
1202   consistently support a single interpretation of the `LocalOnly` parameter. Refer to ANNEX B for details.

1203   If the `DeepInheritance` input parameter is `false`, each returned instance shall not include any
1204   properties added by subclasses of the specified class. If it is `true`, no additional filtering is defined.

1205   **DEPRECATION NOTE:** The use of the `IncludeQualifiers` parameter is DEPRECATED and it may
1206   be removed in a future version of this document. The definition of `IncludeQualifiers` is ambiguous
1207   and when this parameter is set to `true`, WBEM clients cannot be assured that any qualifiers will be
1208   returned. A WBEM client should always set this parameter to `false`. To minimize the impact of this
1209   recommendation on WBEM clients, a WBEM server may choose to treat the value of
1210   `IncludeQualifiers` as `false` for all requests. The preferred behavior is to use the class operations to
1211   receive qualifier information and not depend on any qualifiers in this response. If the
1212   `IncludeQualifiers` input parameter is `true`, all qualifiers for the instance, (including qualifiers on the
1213   instance and on any returned properties, shall be included as <QUALIFIER> XML elements in the
1214   response. If it is `false`, no <QUALIFIER> XML elements are present in the returned instance.

1215   DEPRECATION NOTE: In version 1.4 of this document, the `IncludeClassOrigin` parameter is
1216   DEPRECATED. A WBEM server may choose to treat the value of `IncludeClassOrigin` parameter as
1217   false for all requests, otherwise the implementation shall support the original behavior as defined in the
1218   rest of this paragraph. If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN`
1219   attribute shall be present on all appropriate elements in each returned Instance. If it is `false`, no
1220   `CLASSORIGIN` attributes are present.

1221   If the `PropertyList` input parameter is not NULL, the members of the array define one or more
1222   property names of the designated class. This definition may include inherited property names or property
1223   names explicitly defined in the designated class. However, it may not include property names added in
1224   subclasses of the designated class. Each returned instance shall not include any properties missing from
1225   this list. Note that `PropertyList` acts as an additional filter on the properties defined by the `LocalOnly`
1226   and `DeepInheritance` input parameters; if `PropertyList` includes a property name that is not in the
1227   set defined by the `LocalOnly` and `DeepInheritance` combination, the element for the property shall
1228   not be included in the returned instances. If `PropertyList` is an empty array, no properties are included
1229   in the returned instances. If `PropertyList` is NULL, no additional filtering is defined.

1230   If `PropertyList` contains duplicate property names, the WBEM server shall ignore the duplicates but
1231   otherwise process the request normally. If `PropertyList` contains property names that are invalid for a
1232   target instance, the WBEM server shall ignore them for that instance but otherwise process the request
1233   normally.

1234 Properties with the NULL value may be omitted from the response, even if the WBEM client has not
1235 requested the exclusion of the property through the `LocalOnly`, `DeepInheritance`, or `PropertyList`
1236 filters. The WBEM client shall interpret such omitted properties as NULL. Note that the WBEM client
1237 cannot make any assumptions about properties omitted as a result of using any `LocalOnly`,
1238 `DeepInheritance`, or `PropertyList` filters.

1239 If EnumerateInstances is successful, the method returns zero or more `<namedInstance>` items
1240 representing named instances that meet the required criteria. These instances shall have all properties
1241 defined in and inherited by their respective classes, reduced by any properties excluded as a result of
1242 using the `LocalOnly`, `DeepInheritance`, or `PropertyList` filters and further reduced by any NULL-
1243 valued properties omitted from the response.

1244 If EnumerateInstances is unsuccessful, this method shall return one of the following status codes, where
1245 the error returned is the first applicable error in the list, starting with the first element and working down.
1246 Any additional method-specific interpretation of the error is enclosed in parentheses.

1247 • CIM_ERR_ACCESS_DENIED

1248 • CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)

1249 • CIM_ERR_INVALID_NAMESPACE

1250 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1251 incorrect parameters)

1252 • CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not
1253 exist.)

1254 • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the specified class and all
1255 of its subclasses, if provided.)

1256 • CIM_ERR_FAILED (Some other unspecified error occurred.)

1257 **5.4.2.12 EnumerateInstanceNames (DEPRECATED)**

1258 The EnumerateInstanceNames operation enumerates the names (model paths) of the instances of a CIM
1259 class in the target namespace, including instances in the class and any subclasses in accordance with
1260 the polymorphic nature of CIM objects:

1261 `<instanceName>`* **EnumerateInstanceNames** (
1262   [IN] `<className>` ClassName
1263)

1264 **DEPRECATION NOTE:** The EnumerateInstanceNames operation has been deprecated in version 1.4 of
1265 this document. Use OpenEnumerateInstancePaths instead (see 5.4.2.24.4).

1266 The `ClassName` input parameter defines the class that is the basis for the enumeration.

1267 If EnumerateInstanceNames is successful, the method returns zero or more `<instanceName>` items
1268 representing instance names (referred to in DSP0004 as a model path) that meet the requested criteria.
1269 The `<instanceName>` items shall specify the class from which the instance is instantiated, not any of its
1270 superclasses. Note that this class may be different from the class specified as input.

1271 If EnumerateInstanceNames is unsuccessful, this method shall return one of the following status codes,
1272 where the error returned is the first applicable error in the list, starting with the first element and working
1273 down. Any additional method-specific interpretation of the error is enclosed in parentheses.

1274 • CIM_ERR_ACCESS_DENIED

1275 • CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)

1276   • CIM_ERR_INVALID_NAMESPACE

1277   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1278   incorrect parameters)

1279   • CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not
1280   exist.)

1281   • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the specified class and all
1282   of its subclasses, if provided.)

1283   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1284   **5.4.2.13 ExecQuery (DEPRECATED)**

1285   The ExecQuery operation executes a query against the target namespace:

1286   <object>* **ExecQuery** (
1287   [IN] string QueryLanguage,
1288   [IN] string Query
1289   )

1290   **DEPRECATION NOTE:** The ExecQuery operation has been deprecated in version 1.4 of this document.
1291   Use OpenQueryInstances instead (see 5.4.2.24.14).

1292   The QueryLanguage input parameter defines the query language in which the query parameter is
1293   expressed.

1294   The Query input parameter defines the query to be executed. The results of the query shall be
1295   constrained to contain only CIM classes that exist in the target namespace or CIM instances whose
1296   classes exist in the target namespace. Note that any instances in the result set may or may not exist in
1297   any namespace. Note that for query languages supporting select-lists and from-clauses, this implies that
1298   all select-list entries resolve to disjoint properties exposed by one CIM class named in the from-clause.
1299   This rule does not prevent such queries from using joins.

1300   Neither the query language nor the format of the query is defined by this document. It is anticipated that
1301   query languages will be submitted to the DMTF as separate proposals.

1302   WBEM servers can declare which query languages they support (if any) using a mechanism defined in
1303   7.5.

1304   If ExecQuery is successful, the method returns zero or more <object> items representing CIM classes
1305   or instances that correspond to the results of the query.

1306   If ExecQuery is unsuccessful, the method shall return one of the following status codes, where the error
1307   returned is the first applicable error in the list, starting with the first element and working down. Any
1308   additional method-specific interpretation of the error is enclosed in parentheses.

1309   • CIM_ERR_ACCESS_DENIED

1310   • CIM_ERR_NOT_SUPPORTED

1311   • CIM_ERR_INVALID_NAMESPACE

1312   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1313   incorrect parameters)

1314   • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested query language is not
1315   recognized.)

1316   • CIM_ERR_INVALID_QUERY (The query is not a valid query in the specified query language.)

1317   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1318   **5.4.2.14 Associators (PARTLY DEPRECATED)**

1319   The Associators operation enumerates CIM objects (classes or instances) associated with a particular
1320   source CIM object:

```
1321        <objectWithPath>* Associators (
1322            [IN] <objectName> ObjectName,
1323            [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
1324            [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1325            [IN,OPTIONAL,NULL] string Role = NULL,
1326            [IN,OPTIONAL,NULL] string ResultRole = NULL,
1327            [IN,OPTIONAL] boolean IncludeQualifiers = false,      (DEPRECATED)
1328            [IN,OPTIONAL] boolean IncludeClassOrigin = false,     (DEPRECATED)
1329            [IN,OPTIONAL,NULL] string PropertyList [] = NULL
1330        )
```

1331   **DEPRECATION NOTE:** The Associators operation for instances has been deprecated in version 1.4 of
1332   this document. Use OpenAssociatorInstances instead (see 5.4.2.24.7). The Associators operation for
1333   classes remains undeprecated.

1334   The `ObjectName` input parameter defines the source CIM object whose associated objects are to be
1335   returned. This may be either a class name or instance name (model path).

1336   The `AssocClass` input parameter, if not `NULL`, shall be a valid CIM association class name. It acts as a
1337   filter on the returned set of objects by mandating that each returned object shall be associated to the
1338   source object through an instance of this class or one of its subclasses.

1339   The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the
1340   returned set of objects by mandating that each returned object shall be either an instance of this class (or
1341   one of its subclasses) or be this class (or one of its subclasses).

1342   The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned
1343   set of objects by mandating that each returned object shall be associated with the source object through
1344   an association in which the source object plays the specified role. That is, the name of the property in the
1345   association class that refers to the source object shall match the value of this parameter.

1346   The `ResultRole` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the
1347   returned set of objects by mandating that each returned object shall be associated to the source object
1348   through an association in which the returned object plays the specified role. That is, the name of the
1349   property in the association class that refers to the returned object shall match the value of this parameter.

1350   **DEPRECATION NOTE:** The use of the `IncludeQualifiers` parameter is DEPRECATED and it may
1351   be removed in a future version of this document. The preferred behavior is to use the class operations to
1352   receive qualifier information and not depend on any qualifiers in this response. If `IncludeQualifiers`
1353   is `true`, all qualifiers for each object (including qualifiers on the object and on any returned properties)
1354   shall be included as <QUALIFIER> XML elements in the response. If it is `false`, no <QUALIFIER> XML
1355   elements are present.

1356   DEPRECATION NOTE: In version 1.4 of this document, the `IncludeClassOrigin` parameter is
1357   DEPRECATED for instances. A WBEM server may choose to treat the value of `IncludeClassOrigin`
1358   parameter as false for all instance requests, otherwise the implementation shall support the original
1359   behavior as defined in the rest of this paragraph. If the `IncludeClassOrigin` input parameter is `true`,
1360   the `CLASSORIGIN` attribute shall be present on all appropriate elements in each returned object. If it is
1361   `false`, no `CLASSORIGIN` attributes are present.

1362   If the `PropertyList` input parameter is not NULL, the members of the array define one or more
1363   property names. Each returned object shall not include any properties missing from this list. If
1364   `PropertyList` is an empty array, no properties are included in each returned object. If it is NULL, no
1365   additional filtering is defined.

1366   If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise
1367   process the request normally. If `PropertyList` contains property names that are invalid for a target
1368   object, the WBEM server shall ignore them for that object but otherwise process the request
1369   normally.Clients should not explicitly specify properties in the `PropertyList` parameter unless they
1370   specify a non-NULL value for the `ResultClass` parameter.

1371   If instances are returned, properties with the NULL value may be omitted from the response, even if the
1372   WBEM client has not requested the exclusion of the through the `PropertyList` filter. The WBEM client
1373   shall interpret such omitted properties as NULL. Note that the WBEM client cannot make any
1374   assumptions about properties omitted as a result of using the `PropertyList` filter. If classes are
1375   returned, the WBEM server cannot make this choice, and only the WBEM client can cause properties to
1376   be excluded by using the `PropertyList` filter.

1377   If Associators is successful, the method returns zero or more `<objectWithPath>` items representing
1378   CIM classes or instances meeting the requested criteria. Because it is possible for CIM objects from
1379   different hosts or namespaces to be associated, each returned object includes location information. If the
1380   `ObjectName` refers to a class, then classes are returned. These classes shall have all CIM elements
1381   (properties, methods, and qualifiers) defined in and inherited by that class, reduced by any properties
1382   excluded as a result of using the `PropertyList` filter. If the `ObjectName` refers to an instance, then
1383   instances are returned. These instances shall have all properties defined in and inherited by its class,
1384   reduced by any properties excluded as a result of using the `PropertyList` filter and further reduced by
1385   any NULL valued properties omitted from the response.

1386   If Associators is unsuccessful, this method shall return one of the following status codes, where the error
1387   returned is the first applicable error in the list, starting with the first element and working down. Any
1388   additional method-specific interpretation of the error is enclosed in parentheses.

1389   •   CIM_ERR_ACCESS_DENIED

1390   •   CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)

1391   •   CIM_ERR_INVALID_NAMESPACE

1392   •   CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1393   incorrect parameters)

1394   •   CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1395   instance, if provided.)

1396   •   CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
1397   unspecified error occurred.)

5.4.2.15   AssociatorNames (PARTLY DEPRECATED)

1399   The AssociatorNames operation enumerates the names of CIM Objects (classes or instances) that are
1400   associated with a particular source CIM object:

```
1401    <objectPath>* AssociatorNames (
1402        [IN] <objectName> ObjectName,
1403        [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
1404        [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1405        [IN,OPTIONAL,NULL] string Role = NULL,
1406        [IN,OPTIONAL,NULL] string ResultRole = NULL
```

1407   )

1408   **DEPRECATION NOTE:** The AssociatorNames operation has been deprecated in version 1.4 of this
1409   document. Use OpenAssociatorInstancePaths instead (see 5.4.2.24.8). The AssociatorNames operation
1410   for classes remains undeprecated.

1411   The `ObjectName` input parameter defines the source CIM object whose associated names are to be
1412   returned. This is either a class or instance name (model path).

1413   The `AssocClass` input parameter, if not `NULL`, shall be a valid CIM association class name. It acts as a
1414   filter on the returned set of names by mandating that each returned name identify an object that shall be
1415   associated to the source object through an instance of this class or one of its subclasses.

1416   The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the
1417   returned set of names by mandating that each returned name identify an object that shall be either an
1418   instance of this class (or one of its subclasses) or be this class (or one of its subclasses).

1419   The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned
1420   set of names by mandating that each returned name identify an object that shall be associated to the
1421   source object through an association in which the source object plays the specified role. That is, the
1422   name of the property in the association class that refers to the source object shall match the value of this
1423   parameter.

1424   The `ResultRole` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the
1425   returned set of names by mandating that each returned name identify an object that shall be associated
1426   to the source object through an association in which the named returned object plays the specified role.
1427   That is, the name of the property in the association class that refers to the returned object shall match the
1428   value of this parameter.

1429   If AssociatorNames is successful, the method returns zero or more `<objectPath>` items representing
1430   CIM class paths or instance paths meeting the requested criteria. Because CIM objects from different
1431   hosts or namespaces can be associated, each returned object includes location information. If the
1432   `ObjectName` refers to a class path, then class paths are returned. Otherwise, the `ObjectName` refers to
1433   an instance path, and instance paths are returned.

1434   If AssociatorNames is unsuccessful, one of the following status codes shall be returned by this method,
1435   where the first applicable error in the list (starting with the first element of the list, and working down) is
1436   the error returned. Any additional method-specific interpretation of the error is given in parentheses.

1437   - CIM_ERR_ACCESS_DENIED

1438   - CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)

1439   - CIM_ERR_INVALID_NAMESPACE

1440   - CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise
1441   incorrect parameters)

1442   - CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1443   instance, if provided.)

1444   - CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
1445   unspecified error occurred.)

1446   **5.4.2.16   References (PARTLY DEPRECATED)**

1447   The References operation enumerates the association objects that refer to a particular target CIM object
1448   (class or instance).

1449   <objectWithPath>* **References** (

```
1450          [IN] <objectName> ObjectName,
1451          [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1452          [IN,OPTIONAL,NULL] string Role = NULL,
1453          [IN,OPTIONAL] boolean IncludeQualifiers = false,     (DEPRECATED)
1454          [IN,OPTIONAL] boolean IncludeClassOrigin = false,    (DEPRECATED)
1455          [IN,OPTIONAL,NULL] string PropertyList [] = NULL
1456      )
```

1457 **DEPRECATION NOTE:** The References operation has been deprecated in version 1.4 of this document.
1458 Use OpenReferenceInstances instead (see 5.4.2.24.5). The References operation for classes remains
1459 undeprecated.

1460 The `ObjectName` input parameter defines the target CIM object whose referring objects are to be
1461 returned. This is either a class or instance name (model path).

1462 The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the
1463 returned set of objects by mandating that each returned object shall be an instance of this class (or one of
1464 its subclasses) or this class (or one of its subclasses).

1465 The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned
1466 set of objects by mandating that each returned object shall refer to the target object through a property
1467 with a name that matches the value of this parameter.

1468 **DEPRECATION NOTE:** The use of the `IncludeQualifiers` parameter is DEPRECATED and it may
1469 be removed in a future version of this document. The preferred behavior is to use the class operations to
1470 receive qualifier information and not depend on any qualifiers in this response. If `IncludeQualifiers`
1471 is `true`, all qualifiers for each object (including qualifiers on the object and on any returned properties)
1472 shall be included as <QUALIFIER> XML elements in the response. If this parameter is `false`, no
1473 <QUALIFIER> XML elements are present in each returned Object.

1474 DEPRECATION NOTE: In version 1.4 of this document, the `IncludeClassOrigin` parameter is
1475 DEPRECATED for instances. A WBEM server may choose to treat the value of `IncludeClassOrigin`
1476 parameter as false for all instance requests, otherwise the implementation shall support the original
1477 behavior as defined in the rest of this paragraph. If the `IncludeClassOrigin` input parameter is `true`,
1478 the `CLASSORIGIN` attribute shall be present on all appropriate elements in each returned object. If it is
1479 `false`, no `CLASSORIGIN` attributes are present.

1480 If the `PropertyList` input parameter is not NULL, the members of the array define one or more
1481 property names. Each returned object shall not include any properties missing from this list. If
1482 `PropertyList` is an empty array, no properties are included in each returned object. If `PropertyList`
1483 is NULL, no additional filtering is defined.

1484 If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise
1485 process the request normally. If `PropertyList` contains property names that are invalid for a target
1486 object, the WBEM server shall ignore them for that object but otherwise process the request normally.

1487 Clients should not explicitly specify properties in the `PropertyList` parameter unless they specify a
1488 non-NULL value for the `ResultClass` parameter.

1489 If instances are returned, properties with the NULL value may be omitted from the response, even if the
1490 WBEM client has not requested the exclusion of the property through the `PropertyList` filter. The
1491 WBEM client must interpret such omitted properties as NULL. Note that the WBEM client cannot make
1492 any assumptions about properties omitted as a result of using the `PropertyList` filter. If classes are
1493 returned, the WBEM server cannot make this choice, and only the WBEM client can cause properties to
1494 be excluded by using the `PropertyList` filter.

1495 If References is successful, the method returns zero or more `<objectWithPath>` items representing
1496 CIM classes or instances meeting the requested criteria. Because CIM objects from different hosts or
1497 namespaces can be associated, each returned object includes location information. If the `ObjectName`
1498 refers to a class, then classes are returned. These classes shall have all CIM elements (properties,
1499 methods, and qualifiers) defined in and inherited by that class, reduced by any properties excluded as a
1500 result of using the `PropertyList` filter. If the `ObjectName` refers to an instance, then instances are
1501 returned. These instances shall have all properties defined in and inherited by their respective classes,
1502 reduced by any properties excluded as a result of using the `PropertyList` filter and further reduced by
1503 any NULL valued properties omitted from the response.

1504 If References is unsuccessful, this method shall return one of the following status codes, where the error
1505 returned is the first applicable error in the list, starting with the first element and working down. Any
1506 additional method-specific interpretation of the error is enclosed in parentheses.

1507 • CIM_ERR_ACCESS_DENIED

1508 • CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)

1509 • CIM_ERR_INVALID_NAMESPACE

1510 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1511 incorrect parameters)

1512 • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1513 instance, if provided.)

1514 • CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
1515 unspecified error occurred.)

1516 **5.4.2.17 ReferenceNames (PARTLY DEPRECATED)**

1517 The ReferenceNames operation enumerates the association objects that refer to a particular target CIM
1518 object (class or instance):

```
1519    <objectPath>* ReferenceNames (
1520       [IN] <objectName> ObjectName,
1521       [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1522       [IN,OPTIONAL,NULL] string Role = NULL
1523    )
```

1524 **DEPRECATION NOTE:** The ReferenceNames operation has been deprecated in version 1.4 of this
1525 document. Use OpenReferenceInstancePaths instead (see 5.4.2.24.6). The ReferenceNames operation
1526 for classes remains undeprecated.

1527 The `ObjectName` input parameter defines the target CIM object with the referring object names to be
1528 returned. It may be either a class or an instance name (model path).

1529 The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the
1530 returned set of object names by mandating that each returned Object Name identify an instance of this
1531 class (or one of its subclasses) or this class (or one of its subclasses).

1532 The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned
1533 set of object names by mandating that each returned object name shall identify an object that refers to the
1534 target instance through a property with a name that matches the value of this parameter.

1535 If ReferenceNames is successful, the method returns zero or more `<objectPath>` items representing
1536 CIM class paths or instance paths meeting the requested criteria. Because CIM objects from different
1537 hosts or namespaces can be associated, each returned object includes location information. If the

1538 `ObjectName` refers to a class path, then class paths are returned. Otherwise, the `ObjectName` refers to
1539 an instance path, and instance paths are returned.

1540 If ReferenceNames is unsuccessful, this method shall return one of the following status codes, where the
1541 error returned is the first applicable error in the list, starting with the first element and working down. Any
1542 additional method-specific interpretation of the error is enclosed in parentheses.

1543 • CIM_ERR_ACCESS_DENIED

1544 • CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)

1545 • CIM_ERR_INVALID_NAMESPACE

1546 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1547   incorrect parameters)

1548 • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1549   instance, if provided.)

1550 • CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
1551   unspecified error occurred.)

1552 **5.4.2.18 GetProperty (DEPRECATED)**

1553 The GetProperty operation retrieves a single property value from a CIM instance in the target
1554 namespace:

1555 `<propertyValue>` **GetProperty** (
1556   [IN] `<instanceName>` InstanceName,
1557   [IN] string PropertyName
1558)

1559 **DEPRECATION NOTE:** The GetProperty operation has been deprecated in version 1.4 of this document.
1560 Use GetInstance instead (see 5.4.2.2).

1561 The `InstanceName` input parameter specifies the name of the instance (model path) from which the
1562 property value is requested.

1563 The `PropertyName` input parameter specifies the name of the property with the value to be returned.

1564 If GetProperty is successful, the return value specifies the value of the requested property. If the value is
1565 NULL, no element is returned.

1566 If GetProperty is unsuccessful, this method shall return one of the following status codes, where the error
1567 returned is the first applicable error in the list, starting with the first element and working down. Any
1568 additional method-specific interpretation of the error is enclosed in parentheses.

1569 • CIM_ERR_ACCESS_DENIED

1570 • CIM_ERR_INVALID_NAMESPACE

1571 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1572   incorrect parameters)

1573 • CIM_ERR_INVALID_CLASS (The CIM class does not exist in the specified namespace.)

1574 • CIM_ERR_NOT_FOUND (The CIM class exists, but the requested CIM instance does not exist
1575   in the specified namespace.)

1576 • CIM_ERR_NO_SUCH_PROPERTY (The CIM instance exists, but the requested property does
1577   not.)

1578 • CIM_ERR_FAILED (Some other unspecified error occurred.)

1579   **5.4.2.19 SetProperty (DEPRECATED)**

1580   The SetProperty operation sets a single property value in a CIM instance in the target namespace:

```
1581    void SetProperty (
1582        [IN] <instanceName> InstanceName,
1583        [IN] string PropertyName,
1584        [IN,OPTIONAL,NULL] <propertyValue> NewValue = NULL
1585    )
```

1586   **DEPRECATION NOTE:** The SetProperty operation has been deprecated in version 1.4 of this document.
1587   Use ModifyInstance instead (see 5.4.2.8).

1588   The `InstanceName` input parameter specifies the name of the instance (model path) with the property
1589   value to be updated.

1590   The `PropertyName` input parameter specifies the name of the property with the value to be updated.

1591   The `NewValue` input parameter specifies the new value for the property (which may be NULL).

1592   If SetProperty is unsuccessful, this method shall return one of the following status codes, where the error
1593   returned is the first applicable error in the list, starting with the first element and working down. Any
1594   additional method-specific interpretation of the error is enclosed in parentheses.

1595   •   CIM_ERR_ACCESS_DENIED

1596   •   CIM_ERR_NOT_SUPPORTED (by the WBEM server for this operation)

1597   •   CIM_ERR_INVALID_NAMESPACE

1598   •   CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1599   incorrect parameters)

1600   •   CIM_ERR_INVALID_CLASS (The CIM class does not exist in the specified namespace.)

1601   •   CIM_ERR_NOT_FOUND (The CIM class exists, but the requested CIM instance does not exist
1602   in the specified namespace.)

1603   •   CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1604   instance, if provided.)

1605   •   CIM_ERR_NO_SUCH_PROPERTY (The CIM instance exists, but the requested property does
1606   not.)

1607   •   CIM_ERR_TYPE_MISMATCH (The supplied value is incompatible with the type of the
1608   property.)

1609   •   CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
1610   unspecified error occurred.)

1611   **5.4.2.20 GetQualifier**

1612   The GetQualifier operation retrieves a single qualifier declaration from the target namespace.

```
1613    <qualifierDecl> GetQualifier (
1614        [IN] string QualifierName
1615    )
```

1616   The `QualifierName` input parameter identifies the qualifier with the declaration to be retrieved.

1617   If GetQualifier is successful, the method returns the qualifier declaration for the named qualifier.

1618 If GetQualifier is unsuccessful, this method shall return one of the following status codes, where the error
1619 returned is the first applicable error in the list, starting with the first element and working down. Any
1620 additional method-specific interpretation of the error is enclosed in parentheses.

1621 • CIM_ERR_ACCESS_DENIED

1622 • CIM_ERR_NOT_SUPPORTED

1623 • CIM_ERR_INVALID_NAMESPACE

1624 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1625 incorrect parameters)

1626 • CIM_ERR_NOT_FOUND (The requested qualifier declaration does not exist.)

1627 • CIM_ERR_FAILED (Some other unspecified error occurred.)

1628 **5.4.2.21 SetQualifier**

1629 The SetQualifier operation creates or updates a single qualifier declaration in the target namespace. If the
1630 qualifier declaration already exists, it is overwritten:

```
1631     void SetQualifier (
1632         [IN] <qualifierDecl> QualifierDeclaration
1633     )
```

1634 The QualifierDeclaration input parameter defines the qualifier declaration to add to the
1635 namespace.

1636 If SetQualifier is successful, the qualifier declaration is added to the target namespace. If a qualifier
1637 declaration with the same qualifier name already exists, the new declaration replaces it.

1638 If SetQualifier is unsuccessful, this method returns one of the following status codes, where the error
1639 returned is the first applicable error in the list, starting with the first element and working down. Any
1640 additional method-specific interpretation of the error is enclosed in parentheses.

1641 • CIM_ERR_ACCESS_DENIED

1642 • CIM_ERR_NOT_SUPPORTED

1643 • CIM_ERR_INVALID_NAMESPACE

1644 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1645 incorrect parameters)

1646 • CIM_ERR_FAILED (Some other unspecified error occurred.)

1647 **5.4.2.22 DeleteQualifier**

1648 The DeleteQualifier operation deletes a single qualifier declaration from the target namespace.

```
1649     void DeleteQualifier (
1650         [IN] string QualifierName
1651     )
```

1652 The QualifierName input parameter identifies the qualifier with the declaration to be deleted.

1653 If DeleteQualifier is successful, the specified qualifier declaration is deleted from the namespace.

1654 If DeleteQualifier is unsuccessful, this method shall return one of the following status codes, where the
1655 error returned is the first applicable error in the list, starting with the first element and working down. Any
1656 additional method-specific interpretation of the error is enclosed in parentheses.

1657   • CIM_ERR_ACCESS_DENIED

1658   • CIM_ERR_NOT_SUPPORTED

1659   • CIM_ERR_INVALID_NAMESPACE

1660   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1661   incorrect parameters)

1662   • CIM_ERR_NOT_FOUND (The requested qualifier declaration does not exist.)

1663   • CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.23 EnumerateQualifiers

1664

1665   The EnumerateQualifiers operation enumerates qualifier declarations from the target namespace.

1666   `<qualifierDecl>`* **EnumerateQualifiers** (
1667   )

1668   If EnumerateQualifiers is successful, the method returns zero or more `<qualifierDecl>` items
1669   representing qualifier declarations.

1670   If EnumerateQualifiers is unsuccessful, this method shall return one of the following status codes, where
1671   the error returned is the first applicable error in the list, starting with the first element and working down.
1672   Any additional method-specific interpretation of the error is enclosed in parentheses.

1673   • CIM_ERR_ACCESS_DENIED

1674   • CIM_ERR_NOT_SUPPORTED

1675   • CIM_ERR_INVALID_NAMESPACE

1676   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1677   incorrect parameters)

1678   • CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24 Pulled Enumeration Operations

1679

1680   This clause defines a set of operations that return CIM instances or instance paths in portions controlled
1681   by the WBEM client. These operations are called *pulled enumerations*. Usually, an enumeration session
1682   is established through an Open operation, and subsequent repeated executions of a Pull operation on the
1683   enumeration session are used to retrieve them. Optionally, the Open operation can also pull a first set of
1684   items.

1685   Pulled enumeration operations consist of the following individual operations:

1686   • Open operations open an enumeration of the following instances or instance paths:

1687   – OpenEnumerateInstances (instances of a class)

1688   – OpenEnumerateInstancePaths (instance paths of instances of a class)

1689   – OpenReferenceInstances (association instances referencing a target instance)

1690   – OpenReferenceInstancePaths (the instance paths of association instances referencing a
1691   target instance)

1692   – OpenAssociatorInstances (instances associated with a source instance)

1693   – OpenAssociatorInstancePaths (the instance paths of instances associated to a source
1694   instance)

1695   – OpenQueryInstances (the rows resulting from a query)

1696   • Pull operations are for the following cases:

1697   – PullInstances (Instances are enumerated, and instance paths are either not available, for
1698   example as in for OpenQueryInstances, or not desired.)

1699   – PullInstancesWithPath (Instances with paths are enumerated.)

1700   – PullInstancePaths (Instance paths are enumerated.)

1701   • Other operations are as follows:

1702   – CloseEnumeration (closes an open enumeration)

1703   – EnumerationCount (estimates the number of items in an open enumeration)

1704   **5.4.2.24.1 Behavioral Rules for Pulled Enumeration Operations**

1705   A central concept of pulled enumeration operations is the "enumeration session," which provides a
1706   context in which the operations perform their work and which determines the set of instances or instance
1707   paths to be returned. To process the operations of an enumeration session, some parameters of the
1708   Open operation need to be maintained as long as the enumeration session is open. In addition, some
1709   state data about where the enumeration session is with regard to instances or instance paths already
1710   returned must be maintained.

1711   From a WBEM client perspective, an enumeration session is an enumeration context value. A successful
1712   Open operation establishes the enumeration session and returns an enumeration context value
1713   representing it. This value is used as an input/output parameter in subsequent Pull operations on that
1714   enumeration session. The enumeration context value shall uniquely identify the open enumeration
1715   session within the target CIM namespace of the Open operation that established the enumeration
1716   session. It is valid for a WBEM server to use NULL as an enumeration context value representing a
1717   closed enumeration session, but a WBEM client shall not rely on that.

1718   Defining the enumeration context value in Pull operations as both an input parameter and an output
1719   parameter allows the WBEM server to change the enumeration context value during the execution of a
1720   pull operation. This ability to change allows different implementation approaches on the WBEM server
1721   side, which are transparent for the WBEM client. Example approaches are as follows:

1722   • Maintain any state data describing the enumeration session internally in the WBEM server. The
1723   enumeration context value does not need to change in subsequent Pull operations. The WBEM
1724   server uses this value only to identify the internal state data for the open enumeration session. It
1725   does not use the value to store any state data. A variation of this approach is to hand back
1726   modified enumeration context values for additional WBEM server-side sequence checking.

1727   • Maintain any state data describing the enumeration session only on the WBEM client side. All
1728   state data is stored in the enumeration context value, and the WBEM server does not maintain
1729   any state data about the enumeration session, essentially being completely stateless with
1730   regard to the enumeration session.

1731   • A combination of the two previous approaches.

1732   A WBEM server may support keeping enumeration sessions open across connection terminations and
1733   shutdowns of the server. Objects may be created, deleted, or modified concurrently with an enumeration
1734   session that involves these objects. Such changes may or may not be reflected in the enumeration set.
1735   Therefore, there is no guarantee to the WBEM client that the enumeration set represents a consistent
1736   snapshot of its instances at a point in time. However, the WBEM server should make a best effort attempt
1737   for the returned enumeration set to represent a consistent snapshot of its instances at a point in time. The
1738   order of instances in the enumeration set is undefined.

1739   This document does not restrict the number of enumeration sessions that can be established or executed
1740   concurrently in the same WBEM server or client. This remains true even if the enumeration sets of such
1741   concurrently established enumeration sessions contain the same instances.

1742 Except for CloseEnumeration, all operations on a particular enumeration session shall be executed
1743 sequentially. An enumeration session can be open or closed. It is considered open if operations using its
1744 enumeration context value as an input parameter can be executed successfully. It is opened by the
1745 successful completion of an Open operation and closed by one of the following events:

1746 • Successful completion of a CloseEnumeration operation

1747 • Successful completion of an open or pull operation with the `EndOfSequence` output parameter
1748 set to `true`

1749 • Unsuccessful completion of a pull operation when `ContinueOnError` is not requested

1750 • WBEM server-side decision to close the enumeration session based upon an operation timeout

1751 • WBEM server-side decision to close an enumeration session during an operation on that
1752 enumeration session based upon exceeding server limits

1753 A conformant WBEM server may support closure of enumeration sessions based upon exceeding server
1754 limits. Example situations for such a decision are:

1755 • Pull operations with no objects requested that are repeated with a high frequency on the same
1756 enumeration session

1757 • EnumerationCount operations repeated with a high frequency on the same enumeration
1758 session

1759 A mechanism by which WBEM servers can declare support for closure of enumeration sessions based
1760 upon exceeding server limits is defined in 7.5. If a WBEM server supports such closure of enumeration
1761 sessions, it shall make the decision to close during an operation on that enumeration session. There is no
1762 way to indicate the reason for the closure if the decision is made elsewhere. If a WBEM server closes an
1763 enumeration session based upon exceeding server limits, it shall return failure on the operation on that
1764 enumeration session with the status code CIM_ERR_SERVER_LIMITS_EXCEEDED.

5.4.2.24.2 Common Parameters for the Open Operations

1766 This clause defines commonly used parameters for the Open operations. The description of the individual
1767 Open operations references these parameters as appropriate. Note that not every Open operation uses
1768 every one of these common parameters:

1769 • `EnumerationContext`

1770 – This output parameter is the enumeration context value representing the enumeration
1771 session. If the `EndOfSequence` is `true`, the `EnumerationContext` value may be NULL.

1772 – The representation of an enumeration context value uses a string type. In version 1.3 of
1773 this document, enumeration context values were represented using the
1774 ENUMERATIONCONTEXT XML element. The representation was changed to using a
1775 string type in version 1.4 of this document, because it had turned out that all known
1776 implementations had implemented the enumeration context value using a string type.

1777 • `EndOfSequence`

1778 – This output parameter indicates to the WBEM client whether the enumeration session is
1779 exhausted. If `EndOfSequence` is `true` upon successful completion of an Open operation,
1780 no more instances are available and the WBEM server closes the enumeration session,
1781 releasing any allocated resources related to the enumeration session. If the enumeration
1782 set is empty, it is valid for a WBEM server to set `EndOfSequence` to `true`, even if
1783 `MaxObjectCount` is 0. In this case, the enumeration session is closed upon successful
1784 completion of the Open operation. If `EndOfSequence` is `false`, additional instances may
1785 be available and the WBEM server shall not close the enumeration session.

- `IncludeClassOrigin` **(DEPRECATED)**

  – DEPRECATION NOTE: In version 1.4 of this document, the `IncludeClassOrigin` parameter is DEPRECATED. A WBEM server may choose to treat the value of `IncludeClassOrigin` parameter as false for all requests, otherwise the implementation shall support the original behavior as defined in the rest of this paragraph. 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.

1832　　　　　If the query does not satisfy these rules or if the query is invalid according to the definition
1833　　　　　of the query language, the WBEM server shall return an error with status code
1834　　　　　CIM_ERR_INVALID_QUERY. The Filter Query Language (see DSP0212) automatically
1835　　　　　satisfies these rules.

1836　　● `OperationTimeout`

1837　　　– This input parameter determines the minimum time the WBEM server shall maintain the
1838　　　　open enumeration session after the last Open or Pull operation (unless the enumeration
1839　　　　session is closed during the last operation). If the operation timeout is exceeded, the
1840　　　　WBEM server may close the enumeration session at any time, releasing any resources
1841　　　　allocated to the enumeration session.

1842　　　– An `OperationTimeout` of 0 means that there is no operation timeout. That is, the
1843　　　　enumeration session is never closed based on time.

1844　　　– If `OperationTimeout` is NULL, the WBEM server shall choose an operation timeout.

1845　　　– All other values for `OperationTimeout` specify the operation timeout in seconds.

1846　　　– A WBEM server may restrict the set of allowable values for `OperationTimeout`.
1847　　　　Specifically, the WBEM server may not allow 0 (no timeout). If the specified value is not an
1848　　　　allowable value, the WBEM server shall return failure with the status code
1849　　　　CIM_ERR_INVALID_OPERATION_TIMEOUT. A mechanism by which WBEM servers can
1850　　　　declare the allowable values for `OperationTimeout` is defined in 7.5.

1851　　● `ContinueOnError`

1852　　　– This input parameter, if true, requests a continuation on error, which is the ability to
1853　　　　resume an enumeration session successfully after a Pull operation returns an error. A
1854　　　　mechanism by which conformant WBEM servers can declare support for continuation on
1855　　　　error is defined in 7.5.

1856　　　– If a WBEM server does not support continuation on error and `ContinueOnError` is `true`,
1857　　　　it shall return a failure with the status code
1858　　　　CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED.

1859　　　– If a WBEM server supports continuation on error and `ContinueOnError` is `true`, the
1860　　　　enumeration session shall remain open when a Pull operation fails, and any subsequent
1861　　　　successful Pull operations shall return the set of instances or instance paths that would
1862　　　　have been returned if the failing Pull operations were successful. This behavior is subject
1863　　　　to the consistency rules defined for pulled enumerations. If `ContinueOnError` is `false`,
1864　　　　the enumeration session shall be closed when a Pull operation returns a failure.

1865　　● `MaxObjectCount`

1866　　　– This input parameter defines the maximum number of instances or instance paths that this
1867　　　　Open operation can return. Any uint32 number is valid, including 0. The WBEM server may
1868　　　　deliver any number of instances or instance paths up to `MaxObjectCount` but shall not
1869　　　　deliver more than `MaxObjectCount` elements. A conformant WBEM server
1870　　　　implementation may choose to never return any instances or instance paths during an
1871　　　　Open operation, regardless of the value of `MaxObjectCount`. Note that a WBEM client
1872　　　　can use a `MaxObjectCount` value of 0 to specify that it does not want to retrieve any
1873　　　　instances in the Open operation.

1874　　● Return Value (array of enumerated elements)

1875　　　– The return value of a successful Open operation is an array of enumerated elements with a
1876　　　　number of entries from 0 up to a maximum defined by `MaxObjectCount`. These entries
1877　　　　meet the criteria defined in the Open operation. Note that returning no entries in the array

1878   does not imply that the enumeration session is exhausted. Only the `EndOfSequence`
1879   output parameter indicates whether the enumeration session is exhausted.

5.4.2.24.3 OpenEnumerateInstances

1881   The OpenEnumerateInstances operation establishes and opens an enumeration session of the instances
1882   of a CIM class (including instances of its subclasses) in the target namespace. Optionally, it retrieves a
1883   first set of instances.

1884   <instanceWithPath>* **OpenEnumerateInstances** (
1885   [OUT] string EnumerationContext,
1886   [OUT] Boolean EndOfSequence,
1887   [IN] <className> ClassName,
1888   [IN,OPTIONAL] boolean DeepInheritance = true,
1889   [IN,OPTIONAL] boolean IncludeClassOrigin = false,   **(DEPRECATED)**
1890   [IN,OPTIONAL,NULL] string PropertyList [] = NULL,
1891   [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
1892   [IN,OPTIONAL,NULL] string FilterQuery = NULL,
1893   [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
1894   [IN,OPTIONAL] Boolean ContinueOnError = false,
1895   [IN,OPTIONAL] uint32 MaxObjectCount = 0
1896   )

1897   The OpenEnumerateInstances operation shall comply with the behavior defined in 5.4.2.24.1.

1898   The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

1899   The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

1900   The `ClassName` input parameter defines the class that is the basis for the enumeration. The enumeration
1901   set shall consist of all instances of that specified class, including any instances of any of its subclasses, in
1902   accordance with the polymorphic nature of CIM objects.

1903   The `DeepInheritance` input parameter acts as a filter on the properties included in any enumerated
1904   CIM instances. If the `DeepInheritance` input parameter is `true`, all properties of each enumerated
1905   instance of the class shall be present (subject to constraints imposed by the other parameters), including
1906   any added by subclassing the specified class. If `DeepInheritance` is `false`, each enumerated
1907   instance includes only properties defined for the class specified by `ClassName`.

1908   **(DEPRECATED)** The `IncludeClassOrigin` input parameter is defined in 5.4.2.24.2.

1909   The `PropertyList` input parameter acts as a filter on the properties in any enumerated CIM
1910   instances. If `PropertyList` is not NULL, the members of the array define zero or more property names
1911   of the specified class. This array may include inherited property names or property names explicitly
1912   defined in the specified class. However, it shall not include property names defined in subclasses of the
1913   specified class. Each enumerated instance shall not include any properties missing from this list. Note
1914   that `PropertyList` acts as an additional filter on the properties defined by the `DeepInheritance` input
1915   parameter. If `PropertyList` includes a property that is not in the set defined by `DeepInheritance`,
1916   the element for the property shall not be included. If `PropertyList` is an empty array, no properties are
1917   included in the enumerated instances. If `PropertyList` is NULL, no additional filtering is defined.
1918   If `PropertyList` contains duplicate property names, the WBEM server shall ignore them but otherwise
1919   process the request normally. If `PropertyList` contains property names that are invalid for a target
1920   instance, the WBEM server shall ignore them for that instance but otherwise process the request
1921   normally.

1922   The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

1923   The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

1924   The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

1925   The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

1926   If OpenEnumerateInstances is successful, the return value shall be an array of `<instanceWithPath>`
1927   items representing enumerated instances as defined in 5.4.2.24.2.

1928   The PullInstancesWithPath operation shall be used to pull instances for an enumeration session opened
1929   using OpenEnumerateInstances. If any other operation is used to pull instances, the WBEM server shall
1930   return failure with the status code CIM_ERR_FAILED.

1931   If OpenEnumerateInstances is unsuccessful, this operation shall return one of the following status codes,
1932   where the error returned is the first applicable error in the list, starting with the first element and working
1933   down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

1934   • CIM_ERR_ACCESS_DENIED

1935   • CIM_ERR_SERVER_IS_SHUTTING_DOWN

1936   • CIM_ERR_NOT_SUPPORTED

1937   • CIM_ERR_INVALID_NAMESPACE

1938   • CIM_ERR_INVALID_OPERATION_TIMEOUT

1939   • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

1940   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1941   incorrect parameters)

1942   • CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not
1943   exist.)

1944   • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

1945   • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
1946   not recognized.)

1947   • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
1948   language.)

1949   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1950   **5.4.2.24.4 OpenEnumerateInstancePaths**

1951   The OpenEnumerateInstancePaths operation establishes and opens an enumeration session of the
1952   instance paths of the instances of a CIM class (including instances of its subclasses) in the target
1953   namespace. Optionally, it retrieves a first set of instance paths:

```
1954    <instancePath>* OpenEnumerateInstancePaths (
1955       [OUT] string EnumerationContext,
1956       [OUT] Boolean EndOfSequence,
1957       [IN] <className> ClassName,
1958       [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
1959       [IN,OPTIONAL,NULL] string FilterQuery = NULL,
1960       [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
1961       [IN,OPTIONAL] Boolean ContinueOnError = false,
1962       [IN,OPTIONAL] uint32 MaxObjectCount = 0
1963    )
```

1964   The OpenEnumerateInstancePaths operation shall comply with the behavior defined in 5.4.2.24.1.

1965   The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

1966   The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

1967   The `ClassName` input parameter defines the class that is the basis for the enumeration. The
1968   enumeration set shall consist of the instance paths of all instances of the specified class, including any
1969   instances of any of its subclasses, in accordance with the polymorphic nature of CIM objects.

1970   The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

1971   The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

1972   The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

1973   The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

1974   If OpenEnumerateInstancePaths is successful, the return value shall be an array of `<instancePath>`
1975   items representing enumerated instance paths as defined in 5.4.2.24.2.

1976   The PullInstancePaths operation shall be used to pull instances for an enumeration session opened using
1977   OpenEnumerateInstancePaths. If any other operation is used to pull instances, the WBEM server shall
1978   return failure with the status code CIM_ERR_FAILED.

1979   If OpenEnumerateInstancePaths is unsuccessful, this operation shall return one of the following status
1980   codes, where the error returned is the first applicable error in the list, starting with the first element and
1981   working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

1982   • CIM_ERR_ACCESS_DENIED

1983   • CIM_ERR_SERVER_IS_SHUTTING_DOWN

1984   • CIM_ERR_NOT_SUPPORTED

1985   • CIM_ERR_INVALID_NAMESPACE

1986   • CIM_ERR_INVALID_OPERATION_TIMEOUT

1987   • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

1988   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1989   incorrect parameters)

1990   • CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not
1991   exist.)

1992   • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

1993   • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
1994   not recognized.)

1995   • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
1996   language.)

1997   • 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,  (DEPRECATED)
    [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,

(DEPRECATED) 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.

2043   The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

2044   The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

2045   The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

2046   If OpenReferenceInstances is successful, the return value shall be an array of `<instanceWithPath>`
2047   items representing enumerated instances as defined in 5.4.2.24.2.

2048   The PullInstancesWithPath operation shall be used to pull instances for an enumeration session opened
2049   using OpenReferenceInstances. If any other operation is used to pull instances, the WBEM server shall
2050   return failure with the status code CIM_ERR_FAILED.

2051   If OpenReferenceInstances is unsuccessful, this operation shall return one of the following status codes,
2052   where the error returned is the first applicable error in the list, starting with the first element of and working
2053   down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

2054   • CIM_ERR_ACCESS_DENIED

2055   • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2056   • CIM_ERR_NOT_SUPPORTED

2057   • CIM_ERR_INVALID_NAMESPACE

2058   • CIM_ERR_INVALID_OPERATION_TIMEOUT

2059   • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

2060   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise
2061   incorrect parameters)

2062   • CIM_ERR_NOT_FOUND (The target instance was not found.)

2063   • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

2064   • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
2065   not recognized.)

2066   • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
2067   language.)

2068   • CIM_ERR_FAILED (Some other unspecified error occurred.)

2069   **5.4.2.24.6 OpenReferenceInstancePaths**

2070   The OpenReferenceInstancePaths operation establishes and opens an enumeration session of the
2071   instance paths of the association instances that refer to a particular target CIM instance in the target
2072   namespace. Optionally, it retrieves a first set of instance paths.

```
2073    <instancePath>* OpenReferenceInstancePaths (
2074        [OUT] string EnumerationContext,
2075        [OUT] Boolean EndOfSequence,
2076        [IN] <instanceName> InstanceName,
2077        [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
2078        [IN,OPTIONAL,NULL] string Role = NULL,
2079        [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
2080        [IN,OPTIONAL,NULL] string FilterQuery = NULL,
2081        [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
2082        [IN,OPTIONAL] Boolean ContinueOnError = false,
2083        [IN,OPTIONAL] uint32 MaxObjectCount = 0
2084    )
```

2085 The OpenReferenceInstancePaths operation shall comply with the behavior defined in 5.4.2.24.1.

2086 The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

2087 The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

2088 The `InstanceName` input parameter specifies an instance name (model path) that identifies the target
2089 CIM instance with the referring association instances (respectively, their instance paths) to be
2090 enumerated. Unless restricted by any filter parameters of this operation, the enumeration set shall consist
2091 of the instance paths of all association instances that reference the target instance.

2092 The `ResultClass` input parameter, if not NULL, shall be a CIM class name. It acts as a filter on the
2093 enumerated set of instance paths by mandating that each enumerated instance path shall identify an
2094 instance of this class or one of its subclasses. The WBEM server shall not return an error if the
2095 `ResultClass` input parameter value is an invalid class name or if the class does not exist in the target
2096 namespace.

2097 The `Role` input parameter, if not NULL, shall be a property name. It acts as a filter on the enumerated set
2098 of instance paths by mandating that each enumerated instance path shall identify an instance that refers
2099 to the target instance through a property with a name that matches the value of this parameter. The
2100 WBEM server shall not return an error if the `Role` input parameter value is an invalid property name or if
2101 the property does not exist,

2102 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

2103 The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

2104 The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

2105 The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

2106 If OpenReferenceInstancePaths is successful, the return value shall be an array of `<instancePath>`
2107 items representing enumerated instance paths as defined in 5.4.2.24.2.

2108 The PullInstancePaths operation shall be used to pull instances for an enumeration session opened using
2109 OpenReferenceInstancePaths. If any other operation is used to pull instances, the WBEM server shall
2110 return failure with the status code CIM_ERR_FAILED.

2111 If OpenReferenceInstancePaths is unsuccessful, this operation shall return one of the following status
2112 codes, where the error returned is the first applicable error in the list, starting with the first element and
2113 working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

2114 • CIM_ERR_ACCESS_DENIED

2115 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2116 • CIM_ERR_NOT_SUPPORTED

2117 • CIM_ERR_INVALID_NAMESPACE

2118 • CIM_ERR_INVALID_OPERATION_TIMEOUT

2119 • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

2120 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2121 incorrect parameters)

2122 • CIM_ERR_NOT_FOUND (The target instance was not found.)

2123 • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

2124   • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
2125   not recognized.)

2126   • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
2127   language.)

2128   • CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4.2.24.7 OpenAssociatorInstances

2130   The OpenAssociatorInstances operation establishes and opens an enumeration session of the instances
2131   associated with a particular source CIM instance in the target namespace. Optionally, it retrieves a first
2132   set of instances.

```
2133       <instanceWithPath>* OpenAssociatorInstances (
2134           [OUT] string EnumerationContext,
2135           [OUT] Boolean EndOfSequence,
2136           [IN] <instanceName> InstanceName,
2137           [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
2138           [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
2139           [IN,OPTIONAL,NULL] string Role = NULL,
2140           [IN,OPTIONAL,NULL] string ResultRole = NULL,
2141           [IN,OPTIONAL] boolean IncludeClassOrigin = false, (DEPRECATED)
2142           [IN,OPTIONAL,NULL] string PropertyList [] = NULL,
2143           [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
2144           [IN,OPTIONAL,NULL] string FilterQuery = NULL,
2145           [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
2146           [IN,OPTIONAL] Boolean ContinueOnError = false,
2147           [IN,OPTIONAL] uint32 MaxObjectCount = 0
2148       )
```

2149   The OpenAssociatorInstances operation shall comply with the behavior defined in 5.4.2.24.1.

2150   The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

2151   The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

2152   The `InstanceName` input parameter specifies an instance name (model path) that identifies the source
2153   CIM instance with the associated instances to be enumerated. Unless restricted by any filter parameters
2154   of this operation, the enumeration set shall consist of all instances associated with the source instance.

2155   The `AssocClass` input parameter, if not `NULL`, shall be a CIM association class name. It acts as a filter
2156   on the enumerated set of instances by mandating that each enumerated instance shall be associated with
2157   the source instance through an instance of this class or one of its subclasses. The WBEM server shall not
2158   return an error if the `AssocClass` input parameter value is an invalid class name or if the class does not
2159   exist in the target namespace.

2160   The `ResultClass` input parameter, if not `NULL`, must be a CIM class name. It acts as a filter on the
2161   enumerated set of instances by mandating that each enumerated instance shall be an instance of this
2162   class or one of its subclasses. The WBEM server shall not return an error if the `ResultClass` input
2163   parameter value is an invalid class name or if the class does not exist in the target namespace.

2164   The `Role` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the enumerated set
2165   of instances by mandating that each enumerated instance shall be associated with the source instance
2166   through an association in which the source instance plays the specified role. That is, the name of the
2167   property in the association class that refers to the source instance shall match the value of this

2168 parameter. The WBEM server shall not return an error if the `Role` input parameter value is an invalid
2169 property name or if the property does not exist.

2170 The `ResultRole` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the
2171 enumerated set of instances by mandating that each enumerated instance shall be associated with the
2172 source instance through an association in which the enumerated instance plays the specified role. That
2173 is, the name of the property in the association class that refers to the enumerated instance shall match
2174 the value of this parameter. The WBEM server shall not return an error if the `ResultRole` input
2175 parameter value is an invalid property name or if the property does not exist.

2176 **(DEPRECATED)** The `IncludeClassOrigin` input parameter is defined in 5.4.2.24.2.

2177 The `PropertyList` input parameter acts as a filter on the properties included in any enumerated CIM
2178 instances. If `PropertyList` is not NULL, the members of the array define zero or more property names.
2179 Each enumerated instance shall not include any properties missing from this list. If `PropertyList` is an
2180 empty array, no properties are included in each enumerated instance. If `PropertyList` is NULL, all
2181 properties are included in each enumerated instance, subject to the conditions expressed by the other
2182 parameters. If `PropertyList` contains duplicate property names, the WBEM server shall ignore them
2183 but otherwise process the request normally. If `PropertyList` contains property names that are invalid
2184 for a target instance, the WBEM server shall ignore them for that instance but otherwise process the
2185 request normally. WBEM clients should not specify properties in `PropertyList` unless they specify a
2186 non-NULL value for the ResultClass parameter.

2187 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

2188 The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

2189 The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

2190 The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

2191 If OpenAssociatorInstances is successful, the return value shall be an array of `<instanceWithPath>`
2192 items representing enumerated instances as defined in 5.4.2.24.2.

2193 The PullInstancesWithPath operation shall be used to pull instances for an enumeration session opened
2194 using OpenAssociatorInstances. If any other operation is used to pull instances, the WBEM server shall
2195 return failure with the status code CIM_ERR_FAILED.

2196 If OpenAssociatorInstances is unsuccessful, this operation shall return one of the following status codes,
2197 where the error returned is the first applicable error in the list, starting with the first element and working
2198 down. Any additional operation-specific interpretation of the error is given in parentheses.

2199 • CIM_ERR_ACCESS_DENIED

2200 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2201 • CIM_ERR_NOT_SUPPORTED

2202 • CIM_ERR_INVALID_NAMESPACE

2203 • CIM_ERR_INVALID_OPERATION_TIMEOUT

2204 • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

2205 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2206 incorrect parameters)

2207 • CIM_ERR_NOT_FOUND (The source instance was not found.)

2208 • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

2209   • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
2210   not recognized.)

2211   • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
2212   language.)

2213   • CIM_ERR_FAILED (Some other unspecified error occurred.)

2214   **5.4.2.24.8 OpenAssociatorInstancePaths**

2215   The OpenAssociatorInstancePaths operation establishes and opens an enumeration session of the
2216   instance paths of the instances associated with a particular source CIM instance in the target namespace.
2217   Optionally, it retrieves a first set of instance paths.

```
2218      <instancePath>* OpenAssociatorInstancePaths (
2219         [OUT] string EnumerationContext,
2220         [OUT] Boolean EndOfSequence,
2221         [IN] <instanceName> InstanceName,
2222         [IN,OPTIONAL,NULL] <className> AssocClass= NULL,
2223         [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
2224         [IN,OPTIONAL,NULL] string Role = NULL,
2225         [IN,OPTIONAL,NULL] string ResultRole = NULL,
2226         [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
2227         [IN,OPTIONAL,NULL] string FilterQuery = NULL,
2228         [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
2229         [IN,OPTIONAL] Boolean ContinueOnError = false,
2230         [IN,OPTIONAL] uint32 MaxObjectCount = 0
2231      )
```

2232   This operation shall comply with the behavior defined in 5.4.2.24.1.

2233   The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

2234   The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

2235   The `InstanceName` input parameter specifies an instance name (model path) that identifies the source
2236   CIM instance with the associated instances (respectively, their instance paths) to be enumerated. Unless
2237   restricted by any filter parameters of this operation, the enumeration set shall consist of the instance
2238   paths of all instances associated with the source instance.

2239   The `AssocClass` input parameter, if not `NULL`, shall be a CIM association class name. It acts as a filter
2240   on the enumerated set of instance paths by mandating that each instance path identify an instance that
2241   shall be associated with the source instance through an instance of this class or one of its subclasses.
2242   The WBEM server shall not return an error if the `AssocClass` input parameter value is an invalid class
2243   name or if the class does not exist in the target namespace.

2244   The `ResultClass` input parameter, if not NULL, shall be a CIM class name. It acts as a filter on the
2245   enumerated set of instance paths by mandating that each instance path identify an instance that shall be
2246   an instance of this class or one of its subclasses. The WBEM server shall not return an error if the
2247   `ResultClass` input parameter value is an invalid class name or if the class does not exist in the target
2248   namespace.

2249   The `Role` input parameter, if not NULL, shall be a property name. It acts as a filter on the enumerated set
2250   of instance paths by mandating that each instance path identify an instance that shall be associated with
2251   the source instance through an association in which the source instance plays the specified role. That is,
2252   the name of the property in the association class that refers to the source instance shall match the value

2253   of this parameter. The WBEM server shall not return an error if the `Role` input parameter value is an
2254   invalid property name or if the property does not exist.

2255   The `ResultRole` input parameter, if not NULL, shall be a property name. It acts as a filter on the
2256   enumerated set of instance paths by mandating that each instance path identify an instance that shall be
2257   associated with the source instance through an association in which the instance identified by
2258   the enumerated instance path plays the specified role. That is, the name of the property in the association
2259   class that refers to the instance identified by the enumerated instance path shall match the value of this
2260   parameter. The WBEM server shall not return an error if the `ResultRole` input parameter value is an
2261   invalid property name or if the property does not exist.

2262   The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.4.2.24.2.

2263   The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

2264   The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

2265   The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

2266   If OpenAssociatorInstancePaths is successful, the return value shall be an array of `<instancePath>`
2267   items representing enumerated instance paths as defined in 5.4.2.24.2.

2268   The PullInstancePaths operation shall be used to pull instances for an enumeration session opened using
2269   OpenAssociatorInstancePaths. If any other operation is used to pull instances, the WBEM server shall
2270   return failure with the status code CIM_ERR_FAILED.

2271   If OpenAssociatorInstancePaths is unsuccessful, this operation shall return one of the following status
2272   codes, where the error returned is the first applicable error in the list, starting with the first element and
2273   working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

2274   • CIM_ERR_ACCESS_DENIED

2275   • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2276   • CIM_ERR_NOT_SUPPORTED

2277   • CIM_ERR_INVALID_NAMESPACE

2278   • CIM_ERR_INVALID_OPERATION_TIMEOUT

2279   • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

2280   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2281   incorrect parameters)

2282   • CIM_ERR_NOT_FOUND (The source instance was not found.)

2283   • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

2284   • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
2285   not recognized.)

2286   • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter
2287   language.)

2288   • 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.

2333 **5.4.2.24.10 PullInstancesWithPath**

2334 The PullInstancesWithPath operation retrieves instances including their instance paths from an open
2335 enumeration session represented by an enumeration context value:

```
2336    <instanceWithPath>* PullInstancesWithPath (
2337        [IN,OUT] string EnumerationContext,
2338        [OUT] Boolean EndOfSequence,
2339        [IN] uint32 MaxObjectCount
2340    )
```

2341 The PullInstancesWithPath operation shall comply with the behavior defined in 5.4.2.24.1.

2342 The `EnumerationContext` input/output parameter is defined in 5.4.2.24.9. The enumeration session
2343 shall be established using one of the OpenEnumerateInstances, OpenReferenceInstances, or
2344 OpenAssociatorInstances operations.

2345 The `EndOfSequence` output parameter is defined in 5.4.2.24.9.

2346 The `MaxObjectCount` input parameter is defined in 5.4.2.24.9.

2347 If PullInstancesWithPath is successful, the return value shall be an array of `<instanceWithPath>`
2348 items representing enumerated instances including their instance paths as defined in 5.4.2.24.9.

2349 If PullInstancesWithPath is unsuccessful, this operation shall return one of the following status codes,
2350 where the error returned is the first applicable error in the list, starting with the first element and working
2351 down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

2352 • CIM_ERR_ACCESS_DENIED

2353 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2354 • CIM_ERR_NOT_SUPPORTED

2355 • CIM_ERR_INVALID_NAMESPACE

2356 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2357   incorrect parameters)

2358 • CIM_ERR_INVALID_ENUMERATION_CONTEXT

2359 • CIM_ERR_PULL_HAS_BEEN_ABANDONED

2360 • CIM_ERR_SERVER_LIMITS_EXCEEDED

2361 • CIM_ERR_FAILED (Some other unspecified error occurred.)

2362 **5.4.2.24.11 PullInstancePaths**

2363 The PullInstancePaths operation retrieves instance paths from an open enumeration session represented
2364 by an enumeration context value:

```
2365    <instancePath>* PullInstancePaths (
2366        [IN,OUT] string EnumerationContext,
2367        [OUT] Boolean EndOfSequence,
2368        [IN] uint32 MaxObjectCount
2369    )
```

2370 The PullInstancePaths operation shall comply with the behavior defined in 5.4.2.24.1.

2371   The `EnumerationContext` input/output parameter is defined in 5.4.2.24.9. The enumeration session
2372   shall have been established using one of the OpenEnumerateInstancePaths,
2373   OpenReferenceInstancePaths, or OpenAssociatorInstancePaths operations.

2374   The `EndOfSequence` output parameter is defined in 5.4.2.24.9.

2375   The `MaxObjectCount` input parameter is defined in 5.4.2.24.9.

2376   If PullInstancePaths is successful, the return value shall be an array of `<instancePath>` items
2377   representing enumerated instance paths as defined in 5.4.2.24.9.

2378   If PullInstancePaths is unsuccessful, this operation shall return one of the following status codes, where
2379   the error returned is the first applicable error in the list, starting with the first element and working down.
2380   Any additional operation-specific interpretation of the error is enclosed in parentheses.

2381   •   CIM_ERR_ACCESS_DENIED

2382   •   CIM_ERR_SERVER_IS_SHUTTING_DOWN

2383   •   CIM_ERR_NOT_SUPPORTED

2384   •   CIM_ERR_INVALID_NAMESPACE

2385   •   CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2386   incorrect parameters)

2387   •   CIM_ERR_INVALID_ENUMERATION_CONTEXT

2388   •   CIM_ERR_SERVER_LIMITS_EXCEEDED

2389   •   CIM_ERR_PULL_HAS_BEEN_ABANDONED

2390   •   CIM_ERR_FAILED (Some other unspecified error occurred.)

2391   **5.4.2.24.12CloseEnumeration**

2392   The CloseEnumeration operation closes an open enumeration session, performing an early termination of
2393   an enumeration sequence:

2394   void **CloseEnumeration** (
2395   [IN] string EnumerationContext
2396   )

2397   The `EnumerationContext` parameter is the value representing the enumeration session to be closed.
2398   The enumeration session shall be open and shall be established using one of the Open operations. This
2399   implies that this operation is not to close an enumeration sequence already indicated by
2400   `EndOfSequence` because the sequence has already been closed. The value of the
2401   `EnumerationContext` parameter shall be the value of the `EnumerationContext` output parameter
2402   returned by the previous Pull operation on the enumeration session to be closed.

2403   If CloseEnumeration is successful, the WBEM server shall close the enumeration session represented by
2404   `EnumerationContext`, releasing any allocated resources. Any subsequent use of the
2405   `EnumerationContext` value is unsuccessful.

2406   CloseEnumeration may be executed concurrently with a Pull operation or an EnumerationCount operation
2407   on the same enumeration session. If a WBEM server receives a CloseEnumeration operation request
2408   while it is processing a Pull operation on the same enumeration session, the WBEM server shall attempt
2409   to abandon that Pull operation. If the Pull operation can be abandoned, it shall return a failure with the
2410   status code CIM_ERR_PULL_HAS_BEEN_ABANDONED and the CloseEnumeration operation shall
2411   return success. If the Pull operation cannot be abandoned, it shall proceed as if the CloseEnumeration

2412 operation has not been issued, and the CloseEnumeration operation shall return a failure with the status
2413 code CIM_ERR_PULL_CANNOT_BE_ABANDONED.

2414 If CloseEnumeration is unsuccessful, this operation shall return one of the following status codes, where
2415 the error returned is the first applicable error in the list, starting with the first element and working down.
2416 Any additional operation-specific interpretation of the error is enclosed in parentheses.

2417 • CIM_ERR_ACCESS_DENIED

2418 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2419 • CIM_ERR_NOT_SUPPORTED

2420 • CIM_ERR_INVALID_NAMESPACE

2421 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2422 incorrect parameters)

2423 • CIM_ERR_INVALID_ENUMERATION_CONTEXT

2424 • CIM_ERR_PULL_CANNOT_BE_ABANDONED

2425 • CIM_ERR_FAILED (Some other unspecified error occurred.)

2426 **5.4.2.24.13 EnumerationCount**

2427 The EnumerationCount operation provides an estimated count of the total number of objects in an open
2428 enumeration session represented by an EnumerationContext:

```
2429     uint64 EnumerationCount (
2430         [IN] string EnumerationContext
2431     )
```

2432 The `EnumerationContext` parameter identifies the enumeration session for the EnumerationCount
2433 operation. It shall be established using any of the Open operations and shall be open at the time of the
2434 CloseEnumeration request. A conformant WBEM server may support this operation. A WBEM server that
2435 does not support this operation should respond with the CIM_ERR_NOT_SUPPORTED status.

2436 If EnumerationCount is successful, the operation returns an approximate count of the number of objects
2437 in the enumeration session. This is the number of items remaining to be sent with subsequent Pull
2438 operations. Thus, executing this operation immediately after the open may provide an approximate
2439 estimate of the total number of objects to be returned in the enumeration set. The returned count is only
2440 an estimate of the number of objects to be pulled in the enumeration sequence. This mechanism is
2441 intended to assist WBEM clients in determining the overall size of an enumeration set and the number of
2442 objects remaining in the enumeration session. It should not be used instead of the `EndOfSequence`
2443 parameter to determine the end of an enumeration sequence.

2444 If the WBEM server cannot or will not return an estimate of the number of objects to be returned for the
2445 enumeration context, it may return success and the NULL value.

2446 If EnumerationCount is unsuccessful, this operation shall return one of the following status codes, where
2447 the error returned is the first applicable error in the list, starting with the first element and working down.
2448 Any additional operation-specific interpretation of the error is enclosed in parentheses.

2449 • CIM_ERR_ACCESS_DENIED

2450 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2451 • CIM_ERR_NOT_SUPPORTED

2452 • CIM_ERR_INVALID_NAMESPACE

2453   •   CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2454   incorrect parameters)

2455   •   CIM_ERR_INVALID_ENUMERATION_CONTEXT

2456   •   CIM_ERR_SERVER_LIMITS_EXCEEDED

2457   •   CIM_ERR_FAILED (Some other unspecified error occurred.)

2458   **5.4.2.24.14 OpenQueryInstances**

2459   The OpenQueryInstances operation establishes and opens an enumeration session of the instances of a
2460   CIM class (including instances of its subclasses) in the target namespace. Optionally, it retrieves a first
2461   set of instances:

```
2462    <instance>* OpenQueryInstances (
2463       [IN] string FilterQuery,
2464       [IN] string FilterQueryLanguage,
2465       [IN,OPTIONAL] Boolean ReturnQueryResultClass = false,
2466       [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
2467       [IN,OPTIONAL] Boolean ContinueOnError = false,
2468       [IN,OPTIONAL] uint32 MaxObjectCount = 0,
2469       [OUT, OPTIONAL, NULL] <class> QueryResultClass,
2470       [OUT] string EnumerationContext,
2471       [OUT] Boolean EndOfSequence
2472    )
```

2473   The OpenQueryInstances shall comply with the behavior defined in 5.4.2.24.1.

2474   The `FilterQuery` and `FilterQueryLanguage` input parameters specify the set of enumerated
2475   instances.

2476   `FilterQueryLanguage` shall specify a query language and the value of `FilterQuery` shall be a valid
2477   query in that query language. This document defines neither the query language nor the format of the
2478   query. It is anticipated that query languages will be submitted to the DMTF as separate proposals. A
2479   mechanism by which WBEM servers can declare the query languages they support for filtering in Pulled
2480   enumerations (if any) is defined in 7.5.

2481   The `ReturnQueryResultClass` input parameter controls whether a class definition is returned in
2482   `QueryResultClass`. If it is set to `false`, `QueryResultClass` shall be set to NULL on output. If it is
2483   set to `true`, the value of the `QueryResultClass` on output shall be a class definition that defines the
2484   properties (columns) of each row of the query result.

2485   The `OperationTimeout` input parameter is defined in 5.4.2.24.2.

2486   The `ContinueOnError` input parameter is defined in 5.4.2.24.2.

2487   The `MaxObjectCount` input parameter is defined in 5.4.2.24.2.

2488   The `QueryResultClass` output parameter shall be set to NULL if the `ReturnQueryResultClass`
2489   input parameter is set to `false`. Otherwise, it shall return a class definition where each property of the
2490   class corresponds to one entry of the query select list. The class definition corresponds to one row of the
2491   query result. The class name of this returned class shall be "CIM_QueryResult." This class definition is
2492   valid only in the context of this enumeration.

2493   The `EnumerationContext` output parameter is defined in 5.4.2.24.2.

2494   The `EndOfSequence` output parameter is defined in 5.4.2.24.2.

2495 If OpenQueryInstances is successful, the return value shall be an array of `<instance>` items
2496 representing enumerated instances as defined in 5.4.2.24.2. Such instances are available only in the
2497 context of the enumeration and do not return an instance path. The PullInstancesWithPath operation may
2498 not be used to continue an enumeration started by the OpenQueryInstances operation.

2499 The PullInstances operation shall be used to pull instances for an enumeration session opened using If
2500 OpenQueryInstances. If any other operation is used to pull instances, the WBEM server shall return
2501 failure with the status code CIM_ERR_FAILED.

2502 If OpenQueryInstances is unsuccessful, this operation shall return one of the following status codes,
2503 where the error returned is the first applicable error in the list, starting with the first element and working
2504 down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

2505 • CIM_ERR_ACCESS_DENIED

2506 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2507 • CIM_ERR_NOT_SUPPORTED

2508 • CIM_ERR_INVALID_NAMESPACE

2509 • CIM_ERR_INVALID_OPERATION_TIMEOUT

2510 • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

2511 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2512 incorrect parameters)

2513 • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
2514 not recognized.)

2515 • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
2516 language.)

2517 • CIM_ERR_QUERY_FEATURE_NOT_SUPPORTED (The query requires support for features
2518 that are not supported.)

2519 • CIM_ERR_FAILED (Some other unspecified error occurred.)

2520 **5.4.2.24.15 PullInstances**

2521 The PullInstances operation retrieves instances from an OpenQueryInstances session represented by an
2522 enumeration context value:

```
2523    <instance>* PullInstances (
2524        [IN,OUT] string EnumerationContext,
2525        [OUT] Boolean EndOfSequence,
2526        [IN] uint32 MaxObjectCount
2527    )
```

2528 The PullInstances operation shall comply with the behavior defined in 5.4.2.24.1.

2529 The `EnumerationContext` input/output parameter is defined in 5.4.2.24.9. The enumeration session
2530 shall be established using the OpenQueryInstances operation.

2531 The `EndOfSequence` output parameter is defined in 5.4.2.24.9.

2532 The `MaxObjectCount` input parameter is defined in 5.4.2.24.9.

2533 If PullInstances is successful, the return value shall be an array of `<instance>` items representing
2534 enumerated instances as defined in 5.4.2.24.9.

2535　If PullInstances is unsuccessful, this operation shall return one of the following status codes, where the
2536　error returned is the first applicable error in the list, starting with the first element and working down. Any
2537　additional operation-specific interpretation of the error is enclosed in parentheses.

2538　　　• CIM_ERR_ACCESS_DENIED

2539　　　• CIM_ERR_SERVER_IS_SHUTTING_DOWN

2540　　　• CIM_ERR_NOT_SUPPORTED

2541　　　• CIM_ERR_INVALID_NAMESPACE

2542　　　• CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2543　　　　incorrect parameters)

2544　　　• CIM_ERR_INVALID_ENUMERATION_CONTEXT

2545　　　• CIM_ERR_SERVER_LIMITS_EXCEEDED

2546　　　• CIM_ERR_PULL_HAS_BEEN_ABANDONED

2547　　　• CIM_ERR_FAILED (Some other unspecified error occurred.)

2548　### 5.4.3　Namespace Manipulation Using the CIM_Namespace Class (DEPRECATED)

2549　**DEPRECATION NOTE: This section was deprecated in version 1.4 of this document because it**
2550　**was determined this was outside the scope of this specification. The DMTF WBEM Server profile**
2551　**contains the functionality for manipulating namespaces.**

2552　No intrinsic methods are defined specifically to manipulate namespaces. Namespaces shall be
2553　manipulated using intrinsic methods on the CIM_Namespace class.

2554　#### 5.4.3.1　Namespace Creation

2555　A namespace is created by calling the intrinsic method CreateInstance for the CIM_Namespace class. A
2556　value is specified for the new instance parameter that defines a valid instance of the CIM_Namespace
2557　class and that has a name property that is the desired name of the new namespace.

2558　The proposed definition shall be a correct namespace definition according to DSP0004. Despite the
2559　naming conventions used in the CIM specifications (use of / in namespaces such as root/CIMV2 and
2560　root/CIMV2/test), there is no hierarchy implied among different namespaces. Each namespace is
2561　independent of all others. The namespaces are to be considered flat, and there is no defined behavior for
2562　navigating namespaces.

2563　In creating the new namespace, the WBEM server shall conform to the following rules:

2564　　　• The namespace defined by name property shall not already exist in the WBEM server.

2565　　　• The <LOCALNAMESPACEPATH> defined for the operation defines the namespace in which
2566　　　　the CIM_Namespace instance associated with this new namespace is created.

2567　It is recommended that instances of CIM_Namespace be created in root unless there is a specific reason
2568　to define them in another namespace. The inclusion of a CIM_Namespace instance within a namespace
2569　other than root is allowed.

2570　In addition to creating instances of CIM_Namespace, compliant implementations shall also create an
2571　instance of the association class CIM_NamespaceInManager defining the linking of the namespace
2572　created to the current CIM_ObjectManager.

2573 If CreateInstance is successful, the WBEM server creates the specified namespace. In addition, the
2574 WBEM server shall return information about the namespace as an instance of the class CIM_Namespace
2575 and of returning instances of the association class CIM_NamespaceInManager for each
2576 CIM_Namespace instance created.

5.4.3.2 Namespace Deletion

2578 If the WBEM server supports the CIM_Namespace class, all valid namespaces shall be represented by
2579 an instance of the CIM_Namespace class. A namespace is deleted using the intrinsic method
2580 DeleteInstance to delete the instance of the class CIM_Namespace that represents the namespace. The
2581 namespace to be deleted shall exist.

2582 If DeleteInstance is successful, the WBEM server shall remove the specified CIM_Namespace instance.

2583 If DeleteInstance is unsuccessful, one of the status codes defined for the DeleteInstance operation shall
2584 be returned. A WBEM server may return CIM_ERR_FAILED if a non-empty namespace cannot
2585 successfully be deleted.

5.4.3.3 Manipulation and Query of Namespace Information

2587 The query of namespaces is provided through the following means:

2588 • Query of the CIM_Namespace class on an individual namespace

2589 • Use of the CIM_NamespaceInManager association to link the target CIM_ObjectManager and
2590 the instances of CIM_Namespace representing all namespaces defined in the target
2591 CIM_ObjectManager

5.4.3.4 Use of the __Namespace Pseudo Class (DEPRECATED)

2593 In previous versions of this document, namespaces were manipulated through the pseudo class
2594 __Namespace as follows:

2595 No intrinsic methods are specifically defined for manipulating CIM namespaces. However, modeling a
2596 CIM namespace using class __Namespace, together with the requirement that the root namespace be
2597 supported by all WBEM servers, implies that all namespace operations can be supported.

2598 For example, all child namespaces of a particular namespace are enumerated by calling the intrinsic
2599 method EnumerateInstanceNames against the parent namespace, specifying a value for the ClassName
2600 parameter of __Namespace. A child namespace is created by calling the intrinsic method CreateInstance
2601 against the parent namespace, specifying a value for the NewInstance parameter that defines a valid
2602 instance of the class __Namespace and that has a name property that is the desired name of the new
2603 namespace.

2604 **DEPRECATION NOTE:** The use of the __Namespace class is DEPRECATED. In its place, use the
2605 CIM_Namespace class.

5.4.4 Functional Profiles (DEPRECATED)

2607 **DEPRECATION NOTE: This section was deprecated in version 1.4 of this document and there is
2608 no replacement.**

2609 To establish conformance, this clause partitions the intrinsic methods into functional groups.

2610 Support for a particular group does *not* guarantee that all invocations of a method in that group will
2611 succeed. Rather, the exclusion of a group is a declaration that any attempt to call a method in that group
2612 always returns CIM_ERR_NOT_SUPPORTED.

2613 Mechanisms by which a WBEM server may declare the functional groups that it supports are defined in
2614 7.5.

2615 To limit the number of different profiles that a WBEM server may support, each functional group has a
2616 dependency on another group (with the exception of the Basic Read functional group). If functional group
2617 G_1 has a dependency on functional group G_2, then a WBEM server that supports G_1 shall also support
2618 G_2.

2619 The dependency relation is transitive, so if G_1 depends on G_2, and G_2 depends on G_3, then G_1 depends
2620 on G_3. It is also anti-symmetric, so if G_1 depends on G_2, then G_2 cannot depend on G_1.

2621 Using these rules, Table 3 defines a rooted-directed tree of dependencies with the Basic Read
2622 dependency representing the root node.

2623 For example, a WBEM server that supports the Schema Manipulation functional group shall also support
2624 the Instance Manipulation, Basic Write, and Basic Read.

2625 A WBEM server shall support the Basic Read functional group.

2626   **Table 3 – Root-Directed Tree of Functional Profile Dependencies**

Functional Group	Dependency	Methods
Basic Read	none	GetClass EnumerateClasses EnumerateClassNames GetInstance EnumerateInstances **(DEPRECATED)** EnumerateInstanceNames **(DEPRECATED)** 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 **(PARTLY DEPRECATED)** AssociatorNames **(PARTLY DEPRECATED)** References **(PARTLY DEPRECATED)** ReferenceNames **(PARTLY DEPRECATED)**
Query Execution	Basic Read	ExecQuery
Qualifier Declaration	Schema Manipulation	GetQualifier SetQualifier DeleteQualifier EnumerateQualifiers

5.4.5 Extrinsic Method Invocation

2627

2628 Any WBEM server is assumed to support extrinsic methods, which are defined by the schema supported
2629 by the WBEM server. If a WBEM server does not support extrinsic method invocations, it shall return the
2630 error code CIM_ERR_NOT_SUPPORTED to any request to execute an extrinsic method (subject to the

2631 considerations described in the rest of this clause). This allows a <u>WBEM client</u> to determine that all
2632 attempts to execute extrinsic methods will fail.

2633 If the WBEM server cannot invoke extrinsic methods, it shall return one of the following status codes,
2634 where the error returned is the first applicable error in the list, starting with the first element and working
2635 down. Any additional specific interpretation of the error is enclosed in parentheses.

2636   • CIM_ERR_ACCESS_DENIED

2637   • CIM_ERR_NOT_SUPPORTED (The WBEM server does not support extrinsic method
2638   invocations.)

2639   • CIM_ERR_INVALID_NAMESPACE

2640   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2641   incorrect parameters)

2642   • CIM_ERR_NOT_FOUND (The target CIM class or instance does not exist in the specified
2643   namespace.)

2644   • CIM_ERR_METHOD_NOT_FOUND

2645   • CIM_ERR_METHOD_NOT_AVAILABLE (The WBEM server is unable to honor the invocation
2646   request.)

2647   • CIM_ERR_FAILED (Some other unspecified error occurred.)

2648 ## 5.5   CIM Export Syntax and Semantics

2649 This clause focuses on export methods and their invocation, as well as on functional profiles.

2650 ### 5.5.1   Export Method Invocations

2651 All CIM-XML export message requests defined for the CIM-to-HTTP mapping are invocations of one or
2652 more export methods. Export methods do not operate against CIM namespaces.

2653 An export method call is represented in XML by the <EXPMETHODCALL> element, and the response to
2654 that call is represented by the <EXPMETHODRESPONSE> element.

2655 An input parameter has an IN qualifier with value `true` in the method definition. An output parameter has
2656 an OUT qualifier with value `true` in the method definition. A parameter may be both an input parameter
2657 and an output parameter.

2658 The <EXPMETHODCALL> element names the method to be invoked and supplies any input parameters
2659 to the export method call:

2660   • Each input parameter shall be named using the name assigned in the method definition.

2661   • Input parameters may be supplied in any order.

2662   • Each input parameter of the method, and no others, shall be present in the call unless it is
2663   optional.

2664 The <EXPMETHODRESPONSE> element defines either an <ERROR> or a (possibly optional) return
2665 value and output parameters, which are decorated with the OUT qualifier in the method definition. In the
2666 latter case, the following rules apply:

2667   • Each output parameter shall be named using the name assigned in the method definition.

2668   • Output parameters may be supplied in any order.

2669   • Each output parameter of the method, and no others, shall be present in the response, unless it
2670   is optional.

2671 The method invocation process may be thought of as a two-part process:

2672 • Binding the input parameter values specified as child elements of the <EXPMETHODCALL>
2673 element to the input parameters of the method.

2674 • Attempting to execute the method using the bound input parameters, with one of the following
2675 results:

2676 – If the attempt to call the method is successful, the return value and output parameters are
2677 bound to the child elements of the <EXPMETHODRESPONSE> element.

2678 – If the attempt to call the method is unsuccessful, an error code and (optional) human-
2679 readable description of that code is bound to the <EXPMETHODRESPONSE> element.

2680 **5.5.1.1 Simple Export**

2681 A simple export requires the invocation of a single export method. A simple export request is represented
2682 by a <SIMPLEEXPREQ> element, and a simple export response is represented by a <SIMPLEEXPRSP>
2683 element.

2684 A <SIMPLEEXPREQ> shall contain a <EXPMETHODCALL> element.

2685 **5.5.1.2 Multiple Export**

2686 A multiple export requires the invocation of more than one export method. A multiple export request is
2687 represented by a <MULTIEXPREQ> element, and a multiple export response is represented by a
2688 <MULTIEXPRSP> element.

2689 A <MULTIEXPREQ> (or its respective <MULTIEXPRSP>) element is a sequence of two or more
2690 <SIMPLEEXPREQ> (or its respective <SIMPLEEXPRSP>) elements.

2691 A <MULTIEXPRSP> element shall contain a <SIMPLEEXPRSP> element for every <SIMPLEEXPREQ>
2692 element in the corresponding multiple export response. These <SIMPLEEXPRSP> elements shall be in
2693 the same order as their <SIMPLEEXPREQ> counterparts. The first <SIMPLEEXPRSP> in the response
2694 corresponds to the first <SIMPLEEXPREQ> in the request, and so forth.

2695 Multiple exports conveniently batch the delivery of multiple export method invocations into a single HTTP
2696 message, reducing the number of roundtrips between a WBEM client and a WBEM listener and allowing
2697 the WBEM listener to make certain internal optimizations. Note that multiple exports do not confer any
2698 transactional capabilities in processing the request. For example, the WBEM listener does not have to
2699 guarantee that the constituent export method calls either all failed or all succeeded. The WBEM listener
2700 must only make a "best effort" to process the operation. However, WBEM listeners shall finish processing
2701 each method invocation in a batched message before executing the next method invocation in the batch.
2702 Clients shall recognize that the order of method calls within a batched message is significant.

2703 Not all WBEM listeners support multiple exports. If a WBEM listener does not support multiple exports, it
2704 shall return the status code CIM_ERR_NOT_SUPPORTED.

2705 **5.5.1.3 Status Codes**

2706 This clause defines the status codes and detailed error information that a conforming WBEM listener may
2707 return.

2708 The value of an <ERROR> child element within a <EXPMETHODRESPONSE> element includes the
2709 following parts:

2710 • mandatory status code

2711 • optional human-readable description of the status code

2712 • zero or more CIM_Error instances

2713 The symbolic names defined in Table 4 do not appear on the wire. They are used here solely for
2714 convenient reference to an error in other parts of this document. Not all methods are expected to return
2715 all these status codes.

2716 In addition to returning a status code, a conforming WBEM listener may return zero or more
2717 <INSTANCE> child elements as part of an <ERROR> element. Each <INSTANCE> child element shall
2718 be an instance of CIM_Error, and the value of CIMStatusCode shall comply with the definition of expected
2719 error codes for the CIM-XML export request. A WBEM client may ignore any <INSTANCE> child
2720 elements.

2721 **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.

2722 ## 5.5.2   Export Methods

2723 This clause describes the methods that can be defined within a CIM-XML export message. These
2724 methods operate only on an external data representation of a CIM entity, namespace, or element.
2725 Specifically, export methods do not operate on CIM namespaces or elements. The export method defined
2726 in this document is Export an Indication.

2727 The notation used in the following subclauses to define the signatures of the export methods is a pseudo-
2728 MOF notation that extends the standard MOF BNF (DSP0004) for describing CIM export methods with a
2729 number of pseudo parameter types. The pseudo parameter types are enclosed in angle brackets (< >).

2730 This notation allows parameters to be decorated with pseudo-qualifiers (IN, OPTIONAL, and NULL) to
2731 define their invocation semantics. Note that these qualifiers are for description purposes only within the
2732 scope of this document. In particular, a WBEM client shall not specify them in export method invocations.

2733 This notation uses the IN qualifier for input parameters.

2734 A WBEM client may omit an optional parameter if the required value is the specified default by not
2735 specifying an <EXPPARAMVALUE> element for the parameter. It shall not omit a parameter that is not
2736 optional.

2737 The NULL qualifier indicates parameters with values that may be specified as NULL in an export method
2738 call. A NULL (unassigned) value for a parameter is specified by an <EXPPARAMVALUE> element with
2739 no child element. The WBEM client shall specify a value for parameters without the NULL qualifier by
2740 including a suitable child element for the <EXPPARAMVALUE> element.

2741 All parameters shall be uniquely named and shall correspond to a valid parameter name for that method
2742 as described by this document. The order of the parameters is not significant.

2743 The non-NULL values of export method parameters or return values that are modeled as standard CIM
2744 types (such as string and Boolean, or arrays thereof) are represented as follows:

2745   •   Simple values shall be represented by the <VALUE> child element in an <EXPPARAMVALUE>
2746   element (for export method parameters) or in an <IRETURNVALUE> element (for export
2747   method return values).

2748   • Array values shall be represented by the <VALUE.ARRAY> child element in an
2749   <EXPPARAMVALUE> element (for export method parameters) or in an <IRETURNVALUE>
2750   element (for export method return values).

2751   Table 5 shows how each pseudo-type used by the export methods shall be mapped to an XML element
2752   described in DSP0201 in the context of both a parameter value (child element of <EXPPARAMVALUE>)
2753   and a return value (child element of <IRETURNVALUE>).
2754

2755 **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

2756 ### 5.5.2.1   ExportIndication

2757 The ExportIndication operation exports a single CIM indication to the destination WBEM listener:

```
2758     void ExportIndication (
2759        [IN] <instance> NewIndication
2760     )
```

2761 The `NewIndication` input parameter defines the indication to be exported. The proposed definition
2762 should be a correct instance definition for the underlying CIM indication class according to the CIM
2763 specification.

2764 If ExportIndication is unsuccessful, this method shall return one of the following status codes, where the
2765 error returned is the first applicable error in the list, starting with the first element and working down. Any
2766 additional method-specific interpretation of the error is enclosed in parentheses.

2767 • CIM_ERR_ACCESS_DENIED

2768 • CIM_ERR_NOT_SUPPORTED

2769 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2770 incorrect parameters)

2771 • CIM_ERR_INVALID_CLASS (The CIM class of which this is to be a new instance does not
2772 exist.)
2773 **DEPRECATED:** The use of CIM_ERR_INVALID_CLASS has been deprecated in version 1.4 of
2774 this document because a WBEM listener has no notion about existing classes. Listeners should
2775 not use this status code anymore, and WBEM servers receiving this status code should treat it
2776 like CIM_ERR_FAILED.

2777 • CIM_ERR_FAILED (Some other unspecified error occurred.)

2778 ## 5.5.3   Functional Profiles (DEPRECATED)

2779 **DEPRECATION NOTE: This section was deprecated in version 1.4 of this document and there is
2780 no replacement.**

2781 This clause partitions the export methods into functional groups to establish conformance. See Table 6.

2782 Support for a particular group does not guarantee that all invocations of an export method in that group
2783 will succeed. Rather, the exclusion of a group is a declaration that any attempt to call an export method in
2784 that group always returns CIM_ERR_NOT_SUPPORTED.

2785 The dependency relation is transitive, so if group G_1 depends on G_2, and G_2 depends on G_3, then G_1
2786 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.

2787 <div align="center">**Table 6 – Functional Groups of Export Methods**</div>

Functional Group	Dependency	Method
Indication	None	ExportIndication

2788 6   Encapsulation of CIM-XML Messages

2789 This clause describes how to use CIM-XML messages in HTTP. CIM-XML message requests may be
2790 used with or without the HTTP Extension Framework.

2791 Although CIM-XML messages can be used in combination with a variety of HTTP request methods, this
2792 document defines CIM-XML messages only within HTTP POST requests. (M-POST may be used in place
2793 of POST. For details on how to use CIM-XML messages with the HTTP Extension Framework, see 6.2.)

2794 All CIM-XML message responses are carried in the corresponding HTTP response. In the remaining
2795 discussion, the following terms are used as convenient shorthand for the definitions provided here:

2796 • *CIM-XML operation request.* An HTTP POST request message with an XML entity body that
2797   defines an operation request message.

2798 • *CIM-XML operation response.* An HTTP response message, issued in response to a CIM-XML
2799   operation request, with an entity body that defines an operation response message.

2800 • *CIM-XML export request.* An HTTP POST request message with an XML entity body that
2801   defines an export request message.

2802 • *CIM-XML export response.* An HTTP response message, issued in response to a CIM-XML
2803   export message request, with an entity body that defines an export response message.

2804 • *CIM-XML message request.* An HTTP POST request message with an XML entity body that
2805   defines either an operation request message or an export request message.

2806 • *CIM-XML message response.* An HTTP response message, issued in response to a CIM-XML
2807   message request, with an entity body that defines either an operation response message or an
2808   export response message.

2809 Note that an HTTP response to a CIM request is not always a CIM response. For example, a "505 HTTP
2810 Version Not Supported" response is not a CIM response.

2811 6.1   WBEM clients, WBEM servers, and WBEM listeners

2812 A *CIM product* is any product that can supply and/or consume management information using the CIM
2813 schema. In particular, WBEM clients, WBEM servers, and WBEM listeners are examples of CIM products:

2814 • A *WBEM client* issues CIM-XML operation requests and receives and processes CIM-XML
2815   operation responses.

2816 • A *WBEM server* receives and processes CIM-XML operation requests and issues CIM-XML
2817   operation responses. A WBEM server also issues CIM-XML export requests and receives and
2818   processes CIM-XML export responses.

2819   • A *WBEM listener* is a server that receives and processes CIM-XML export requests and issues
2820   CIM-XML export responses.

2821   Throughout this document, the terms WBEM client, WBEM server, WBEM listener, and CIM product are
2822   used as convenient shorthand to refer to the subset of CIM products that conform to this document.

2823   Note that "WBEM client" (server, listener) was used for the term "WBEM client" (server, listener) before
2824   version 1.4 of this document.

2825   6.2   Use of M-POST

2826   A WBEM client attempting to invoke a CIM-XML message using the HTTP Extension Framework method
2827   "M-POST" shall follow these steps:

2828   • If the M-POST invocation fails with an HTTP status of "501 Not Implemented" or "510 Not
2829   Extended," the client should retry the request using the HTTP method "POST" with the
2830   appropriate modifications (described in 6.2.2).

2831   • If the M-POST invocation fails with an HTTP status of "405 Method Not Allowed," the client
2832   should fail the request.

2833   • For all other status codes, the client shall act in accordance with standard HTTP (see 7.1).

2834   This extended invocation mechanism gives Internet proxies and firewalls greater filtering control and
2835   administrative flexibility over CIM-XML message invocations.

2836   If a client receives a 501 or 510 status in response to an M-POST request, in subsequent invocations to
2837   the same HTTP server, the client may omit the attempt at M-POST invocations for a suitable period. This
2838   omission avoids the need for an extra round trip on each and every method invocation. The details of the
2839   caching strategy employed by the client are outside the scope of this document.

2840   6.2.1   Use of the Ext Header

2841   If a WBEM server or WBEM listener receives a valid M-POST request and has fulfilled all mandatory
2842   extension header declarations in the request, it shall include in the response the "Ext" header defined by
2843   RFC2774. This included header shall be protected by the appropriate Cache-Control directive.

2844   6.2.2   Naming of Extension Headers

2845   In M-POST request messages (and their responses), CIM extension headers shall be declared using the
2846   name space prefix allotted by the "Man" extension header (in accordance with RFC2774) that refers to
2847   the name space "http://www.dmtf.org/cim/mapping/http/v1.0". The full format of the "Man" header
2848   declaration for this document is:

```
2849       Man             = "Man" ":" "http://www.dmtf.org/cim/mapping/http/v1.0"
2850                         ";" "ns" "=" header-prefix
2851
2852       header-prefix = 2*DIGIT
```

2853   This header-prefix should be generated at random on a per-HTTP message basis, and should not
2854   necessarily be a specific number.

2855   In accordance with RFC2774, all POST request messages (and their responses) shall not include such a
2856   mandatory extension declaration. In POST request messages (and their responses), name space
2857   prefixes shall not be used.

2858   EXAMPLE 1:

2859   Using M-POST:

```
2860    M-POST /cimom HTTP/1.1
2861    Man: http://www.dmtf.org./cim/mapping/http/v1.0 ; ns=23
2862    23-CIMOperation: MethodCall
2863    ...
```

2864 EXAMPLE 2:

2865 Using POST:

```
2866    POST /cimom HTTP/1.1
2867    CIMOperation: MethodCall
2868    ...
```

2869 6.3 Extension Headers Defined for CIM-XML Message Requests and Responses

2870 A CIM-XML message contains exactly one CIM-XML operation request, CIM-XML operation response,
2871 CIM-XML export request, or CIM-XML export response. This clause describes the extension headers to
2872 specify CIM-XML message semantics in the HTTP header of a POST message.

2873 Any CIM-XML operation request or CIM-XML operation response shall, and only CIM-XML operation
2874 requests and responses may, include the following CIM extension header:

2875 • CIMOperation

2876 Any CIM-XML operation request shall, and only CIM-XML operation requests may, include one and only
2877 one of the following CIM extension header sets:

2878 • CIMMethod and CIMObject, or

2879 • CIMBatch **(DEPRECATED)**

2880 Any CIM-XML export request or CIM-XML export response shall, and only CIM-XML export requests and
2881 responses may, include the following CIM extension header:

2882 • CIMExport

2883 Any CIM-XML export request shall, and only CIM-XML export requests may, include one and only one of
2884 the following CIM extension headers:

2885 • CIMExportMethod

2886 • CIMExportBatch **(DEPRECATED)**

2887 An HTTP response with an error status code to a CIM-XML message request may include the following
2888 CIM extension header:

2889 • CIMError

2890 All CIM-XML messages may include the following CIM extension header:

2891 • CIMProtocolVersion

2892 6.3.1 Encoding of CIM Element Names within HTTP Headers and Trailers

2893 CIM element (class, property, qualifier, method, or method parameter) names are natively Unicode, and
2894 may use UCS-2 characters unsuitable for inclusion within an HTTP message header or trailer. To encode
2895 CIM element names represented in Unicode to values within HTTP headers or trailers, the following two-
2896 step mapping process shall be used:

2897 • Encode the full Unicode CIM element name using UTF-8.

2898   • Using the ""%" HEX HEX" convention, apply the standard URI [RFC2396, section 2] escaping
2899   mechanism to the resulting string to escape any characters that are unsafe within an HTTP
2900   header or trailer.

2901   In this document, the token CIMIdentifier represents a CIM element name to which this transformation
2902   has been applied.

2903   One characteristic of this mapping is that CIM elements named with an ASCII representation appear in
2904   ASCII in the resulting URL.

2905   EXAMPLES:

2906   • CIM_LogicalElement is unchanged under this transformation.

2907   • The class named using the UCS-2 sequence representing the Hangul characters for the Korean
2908   word "hangugo" (D55C, AD6D, C5B4) becomes

2909   ```
       %ED%95%9C%EA%B5%AD%EC%96%B4=10
       ```

2910       after UTF-8 transformation and escaping all characters with their % HEX HEX equivalent.

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

2912    This clause describes the mapping that shall be applied to represent CIM object paths, as described
2913    within an Operation Request Message using the <LOCALNAMESPACEPATH>, <LOCALCLASSPATH>,
2914    or <LOCALINSTANCEPATH> elements, in a format that is safe for representation within an HTTP header
2915    or trailer.

2916    If the element to be transformed is a <LOCALNAMESPACEPATH>, the algorithm is as follows:

2917    • For the first <NAMESPACE> child element, output the textual content of that element.

2918    • For each subsequent <NAMESPACE> child element, output the forward slash character (/)
2919       followed by the textual content of that <NAMESPACE> element.

2920    If the element to be transformed is a <LOCALCLASSPATH>, the algorithm is as follows:

2921    • Transform the <LOCALNAMESPACEPATH> child element using the rules previously
2922       described, and output a colon character (:).

2923    • Output the value of the NAME attribute of the <CLASSNAME> child element.

2924    If the element to be transformed is a <LOCALINSTANCEPATH>, the algorithm is as follows:

2925    • Transform the <LOCALNAMESPACEPATH> child element using the rules previously
2926       described, and output a colon character (:).

2927    • Output the value of the CLASSNAME attribute of the <INSTANCENAME> child element.

2928    • If there is at least one <KEYBINDING> child element under the <INSTANCENAME> child
2929       element, then for each such child element:

2930       – Output a period character (.) if this is the first <KEYBINDING> child element; otherwise,
2931          output a comma character (,).

2932       – Output the value of the NAME attribute, followed by an equal character (=).

2933       – If there is a <KEYVALUE> child element, output the textual element content of that
2934          element, subject to the following transformation:

2935          • If the VALUETYPE attribute is numeric or Boolean, the output is identical to the
2936             content of the element.

2937          &bull;    If the VALUETYPE attribute is a string, the output is obtained by enclosing the content
2938               of the element in double quote (") characters and escaping any double quote
2939               characters or backslash character within the value with a preceding backslash (\)
2940               character.

2941        &ndash;    If there is a <VALUE.REFERENCE> child element

2942          &bull;    Output a double quote character (").

2943          &bull;    Apply the process recursively to the <CLASSPATH> or <INSTANCEPATH> child
2944               element of the <VALUE.REFERENCE> element, escaping any double quote or
2945               backslash character thereby generated with a preceding backslash (\) character.

2946          &bull;    Output a closing double quote character (").

2947    &bull;    If there is no <KEYBINDING> child element but there is a <KEYVALUE> or
2948        <VALUE.REFERENCE> child element under the <INSTANCENAME> child element, then:

2949        &ndash;    Output an equal character (=).

2950        &ndash;    Output the transformed value of the <KEYVALUE> or <VALUE.REFERENCE> using the
2951             previously-described rules.

2952    &bull;    If there are no <KEYBINDING> child elements or no <KEYVALUE> or <VALUE.REFERENCE>
2953        child element, then indicate a singleton instance by outputting the string "=@" under the
2954        <INSTANCENAME> child element.

2955    Finally, after applying these rules to the <LOCALNAMESPACEPATH>, <LOCALCLASSPATH>, or
2956    <LOCALINSTANCEPATH> element, transform the entire output string into URI-safe format in the
2957    following two-step procedure:

2958    &bull;    Encode the string using UTF-8 [RFC2279] if it is not already in this format.

2959    &bull;    Using the ""%" HEX HEX" convention, apply the standard URI [RFC2396, section 2] escaping
2960        mechanism to the resulting string to escape any characters that are unsafe within an HTTP
2961        header or trailer.

2962    In this document, the token CIMObjectPath represents a <LOCALNAMESPACEPATH>,
2963    <LOCALCLASSPATH>, or <LOCALINSTANCEPATH> element to which the preceding transformation
2964    has been applied.

### 2965    6.3.3   CIMOperation

2966    The CIMOperation header shall be present in all CIM-XML operation request and CIM-XML operation
2967    response messages. It identifies the HTTP message as carrying a CIM-XML operation request or
2968    response.

2969         `CIMOperation = "CIMOperation" ":" ("MethodCall" | "MethodResponse")`

2970    A WBEM client shall include this header, with the value "MethodCall," in all CIM-XML operation requests
2971    that it issues. A WBEM server shall include this header in all CIM-XML operation responses that it issues,
2972    with the value "MethodResponse".

2973    If a WBEM server receives a CIM-XML operation request with this header, but with a missing value or a
2974    value that is not "MethodCall," then it shall fail the request with status "400 Bad Request". The WBEM
2975    server shall include a CIMError header in the response with a value of `unsupported-operation`.

2976    If a WBEM server receives a CIM-XML operation request without this header, it shall not process it as a
2977    CIM-XML operation request. The status code returned by the WBEM server in response to such a
2978    request is outside the scope of this document.

2979    If a WBEM client receives a response to a CIM-XML operation request without this header (or if this
2980    header has a value that is not "MethodResponse"), it should discard the response and take appropriate
2981    measures to publicize that it has received an incorrect response. The details as to how this is done are
2982    outside the scope of this document.

2983    The CIMOperation header affords a simple mechanism by which firewall or proxy administrators can
2984    make global administrative decisions on all CIM operations.

### 6.3.4   CIMExport

2986    The CIMExport header shall be present in all CIM-XML export request and response messages. It
2987    identifies the HTTP message as carrying a CIM export method request or response.

2988        CIMExport = "CIMExport" ":" ("MethodRequest" | "MethodResponse")

2989    A WBEM client shall include this header with the value "MethodRequest" in all CIM-XML export requests
2990    that it issues. A WBEM listener shall include this header in all CIM-XML export responses that it issues,
2991    with the value "MethodResponse".

2992    If a WBEM listener receives a CIM-XML export request with this header, but with a missing value or a
2993    value that is not "MethodRequest", then it shall fail the request with status "400 Bad Request". The
2994    WBEM listener shall include a CIMError header in the response with a value of unsupported-operation.

2995    If a WBEM listener receives a CIM-XML export request without this header, it shall not process it. The
2996    status code returned by the WBEM listener in response to such a request is outside of the scope of this
2997    document.

2998    If a WBEM client receives a response to a CIM-XML export request without this header (or if this header
2999    has a value that is not "MethodResponse"), it should discard the response and take appropriate
3000    measures to publicize that it has received an incorrect response. The details as to how this is done are
3001    outside the scope of this document.

3002    The CIMExport header affords a simple mechanism by which firewall or proxy administrators can make
3003    global administrative decisions on all CIM exports.

### 6.3.5   CIMProtocolVersion

3005    The CIMProtocolVersion header may be present in any CIM-XML message. The header identifies the
3006    version of the CIM operations over the HTTP specification in use by the sending entity.

3007        CIMProtocolVersion = "CIMProtocolVersion" ":" 1*DIGIT "." 1*DIGIT

3008    If the header is omitted, then a value of 1.0 must be assumed.

3009    The major and minor revision numbers must be treated as independent integers.

3010    The CIMProtocolVersion $x_1.y_1$ is less than CIMProtocolVersion $x_2.y_2$ if and only if one of the following
3011    statements is true:

3012        •    $x_1$ is less than $x_2$

3013        •    $x_1$ equals $x_2$, and $y_1$ is less than $y_2$

3014    The CIMProtocolVersion $x_1.y_1$ is greater than CIMProtocolVersion $x_2.y_2$ if and only if one of the following
3015    statements is true:

3016        •    $x_1$ is greater than $x_2$,

3017        •    $x_1$ equals $x_2$, and $y_1$ is greater than $y_2$

3018    A CIMProtocolVersion $x_1.y_1$ is within tolerance of CIMProtocolVersion $x_2.y_2$ if:

3019 • $x_1$ equals $x_2$, and

3020 • $y_1$ is less than or equal to $y_2$

3021 If the CIMProtocolVersion of the CIM-XML message received is within tolerance of the
3022 CIMProtocolVersion supported for a WBEM server or WBEM listener implementation, the receiving
3023 implementation shall accept that CIM-XML message. Equivalent CIMProtocolVersion values between
3024 WBEM server or WBEM listener and the WBEM client shall be accepted. The WBEM server or WBEM
3025 listener implementation may reject a CIM-XML message in all other cases. For information about how
3026 CIM-XML messages are rejected, see 7.3.

3027 Beyond tolerance considerations, the implementation should reject the received CIM-XML message *only*
3028 if the design as defined by the CIMProtocolVersion of the receiving implementation has changed in the
3029 declaration of the API, method parameters, or behavior since the design defined by the
3030 CIMProtocolVersion of the received CIM-XML message.

### 3031 6.3.6  CIMMethod

3032 The CIMMethod header shall be present in any CIM-XML operation request message that contains a
3033 Simple Operation Request.

3034 It shall not be present in any CIM-XML operation response message nor in any CIM-XML operation
3035 request message unless it is a simple operation request. It shall not be present in any CIM-XML export
3036 request or response message.

3037 The header identifies the name of the CIM method to be invoked, encoded in an HTTP-safe
3038 representation. Firewalls and proxies may use this header to carry out routing and forwarding decisions
3039 based on the CIM method to be invoked.

3040 The name of the CIM method within a simple operation request is the value of the NAME attribute of the
3041 <METHODCALL> or <IMETHODCALL> element.

```
3042     CIMMethod  = "CIMMethod" ":" MethodName
3043
3044     MethodName = CIMIdentifier
```

3045 If a WBEM server receives a CIM-XML operation request for which any one of the following statements is
3046 true, then it shall fail the request and return a status of "400 Bad Request". Also, it shall include a
3047 CIMError header in the response with a value of header-mismatch, subject to the considerations
3048 specified in 7.3:

3049 • The CIMMethod header is present, but it has an invalid value.

3050 • The CIMMethod header is not present, but the operation request message is a Simple
3051   Operation Request.

3052 • The CIMMethod header is present, but the operation request message is not a simple operation
3053   request.

3054 • The CIMMethod header is present and the operation request message is a simple operation
3055   request, but the CIMIdentifier value (when unencoded) does not match the unique method
3056   name within the simple operation request.

3057 Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy was
3058 not acting on misleading information when it decided to forward the request based on the content of the
3059 CIMMethod header. Additional securing of HTTP messages against modification in transit (such as the
3060 encryption of the payload or appending of a digital signature thereto) would be required to provide a
3061 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.

3105 The name of the CIM export method within a simple export request is the value of the NAME attribute of
3106 the <EXPMETHODCALL> element.

3107   CIMExportMethod = "CIMExportMethod" ":" ExportMethodName
3108
3109   ExportMethodName = CIMIdentifier

3110 If a WBEM listener receives a CIM-XML export request for which any one of the following statements is
3111 true, then it shall fail the request and return a status of "400 Bad Request". Also, it shall include a
3112 CIMError header in the response with a value of header-mismatch, subject to the considerations specified
3113 in 7.3:

- 3114 • The CIMExportMethod header is present, but it has an invalid value.

- 3115 • The CIMExportMethod header is not present, but the export request message is a simple export
3116   request.

- 3117 • The CIMExportMethod header is present, but the export request message is not a simple export
3118   request.

- 3119 • The CIMExportMethod header is present and the export request message is a simple export
3120   request, but the CIMIdentifier value (when unencoded) does not match the unique method
3121   name within the simple export request.

3122 Note that this verification provides a basic level of assurance that any intermediate firewall or proxy is not
3123 acting on misleading information when it forwards the request based on the content of the
3124 CIMExportMethod header. Additional securing of HTTP messages against modification in transit, such as
3125 encrypting the payload or appending a digital signature to it, would be required to provide a higher degree
3126 of integrity.

6.3.9 CIMBatch (DEPRECATED)

3128 **DEPRECATION NOTE: This section was deprecated in version 1.4 of this document and there is**
3129 **no replacement.**

3130 The CIMBatch header shall be present in any CIM-XML operation request message that contains a
3131 Multiple Operation Request.

3132 This header shall not be present in any CIM-XML operation response message nor in any CIM-XML
3133 operation request message unless it is a multiple operation request. It shall not be present in any CIM-
3134 XML export request or response message.

3135 The CIMBatch header identifies the encapsulated operation request message as containing multiple
3136 method invocations. Firewalls and proxies may use this header to carry out routing and forwarding
3137 decisions for batched CIM method invocations.

3138   CIMBatch = "CIMBatch" ":"

3139 If a WBEM server receives a CIM-XML operation request for which any one of the following statements is
3140 true, then it must fail the request and return a status of "400 Bad Request". Also it must include a
3141 CIMError header in the response with a value of header-mismatch, subject to the considerations
3142 specified in 7.3:

- 3143 • The CIMBatch header is present, but it has an invalid value.

- 3144 • The CIMBatch header is not present, but the operation request message is a multiple operation
3145   request.

- 3146 • The CIMBatch header is present, but the operation request message is not a multiple operation
3147   request.

3148   Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy is not
3149   acting on misleading information when it forwards the request based on the content of the CIMBatch
3150   header. Additional securing of HTTP messages against modification in transit, such as encrypting the
3151   payload or appending a digital signature to it, would be required to provide a higher degree of integrity.

3152   If a WBEM server receives a CIM-XML operation request for which the CIMBatch header is present but
3153   the server does not support multiple operations, then it shall fail the request and return a status of "501
3154   Not Implemented". Firewalls or Proxies may also employ this mechanism to compel a WBEM client to use
3155   simple operation requests rather than multiple operation requests.

3156   A WBEM client that receives a response of "501 Not Implemented" to a multiple operation request should
3157   resubmit that request as a series of simple operation requests.

6.3.10 CIMExportBatch (DEPRECATED)

3159   **DEPRECATION NOTE: This section was deprecated in version 1.4 of this document and there is**
3160   **no replacement.**

3161   The CIMExportBatch header shall be present in any CIM-XML export request message that contains a
3162   multiple export request.

3163   It shall not be present in any CIM-XML operation request or response message. Also, it shall not be
3164   present in any CIM-XML export response message nor in any CIM-XML export request message unless it
3165   is a multiple export request.

3166   The header identifies the encapsulated Export Request Message as containing multiple export method
3167   invocations. Firewalls and proxies may use this header to carry out routing and forwarding decisions for
3168   batched CIM Export method invocations.

3169   CIMExportBatch = "CIMExportBatch" ":"

3170   If a WBEM listener receives a CIM-XML export request for which any one of the following statements is
3171   true, then it must fail the request and return a status of "400 Bad Request". Also, it must include a
3172   CIMError header in the response with a value of header-mismatch, subject to the considerations specified
3173   in Errors:

3174   - The CIMExportBatch header is present, but it has an invalid value.

3175   - The CIMExportBatch header is not present, but the export request message is a multiple export
3176   request.

3177   - The CIMExportBatch header is present, but the export request message is not a multiple export
3178   request.

3179   Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy is not
3180   acting on misleading information when it forwards the request based on the content of the
3181   CIMExportBatch header. Additional securing of HTTP messages against modification in transit, such as
3182   encrypting the payload or appending a digital signature to it, would be required to provide a higher degree
3183   of integrity.

3184   If a WBEM listener receives a CIM-XML export request for which the CIMExportBatch header is present,
3185   but the WBEM listener does not support multiple exports, then it shall fail the request and return a status
3186   of "501 Not Implemented". Firewalls or Proxies may also employ this mechanism to compel a WBEM
3187   client to use simple rather than multiple export requests.

3188   A WBEM client that receives a response of "501 Not Implemented" to a multiple export request should
3189   resubmit that request as a series of simple export requests.

3190 6.3.11 CIMError

3191 The CIMError header may be present in any HTTP response to a CIM-XML message request that is not a
3192 CIM-XML message response.

3193 It shall not be present in any CIM-XML message response or in any CIM-XML message request.

3194 The CIMError header provides further CIM-specific diagnostic information if the WBEM server or WBEM
3195 listener encounters a fundamental error during processing of the CIM-XML operation request and is
3196 intended to assist clients to further disambiguate errors with the same HTTP status code:

```
3197        CIMError  = "CIMError" ":" cim-error
3198
3199        cim-error = "unsupported-protocol-version" |
3200            "multiple-requests-unsupported" |
3201            "unsupported-cim-version" |
3202            "unsupported-dtd-version" |
3203            "request-not-valid" |
3204            "request-not-well-formed" |
3205            "request-not-loosely-valid" |
3206            "header-mismatch" |
3207            "unsupported-operation"
```

3208 6.3.12 CIMRoleAuthenticate

3209 A WBEM server may return a CIMRoleAuthenticate header as part of the 401 Unauthorized response
3210 along with the WWW-Authenticate header. The CIMRoleAuthenticate header must meet the challenge of
3211 indicating the WBEM server policy on role credentials.

```
3212      challenge = "credentialrequired" | "credentialoptional" | "credentialnotrequired"
```

3213 • A challenge of `credentialrequired` indicates that the WBEM server requires that a WBEM
3214   client must present a credential if it seeks to assume a role.

3215 • A challenge of `credentialoptional` indicates that the credential is optional. If a credential is
3216   not sent, the WBEM server allows the role assumption if it is permitted for the given user.
3217   However, certain operations that require the role credential may not succeed.

3218 • A challenge of `credentialnotrequired` indicates that no credential is required to assume
3219   the role.

3220 Absence of the CIMRoleAuthenticate header indicates that the WBEM server does not support role
3221 assumption. A WBEM client should handle each of these cases appropriately.

3222 The challenge does not contain any authorization scheme, realm, or other information. A WBEM client
3223 should extract this information from the WWW-Authenticate header. This implies that for any given
3224 request, the role credentials should use the same scheme as those required for the user credentials.

3225 A WBEM server allows role assumption to succeed only if the user is allowed to assume the role.
3226 Therefore, even if appropriate credentials are presented, role assumption can fail. If either the user
3227 authentication or role assumption fails, the entire authentication operation fails.

3228 To maintain backward compatibility, a WBEM server that supports role assumption must allow user
3229 authentication even if no role is specified.

3230 6.3.13 CIMRoleAuthorization

3231 The CIMRoleAuthorization header is supplied along with the normal authorization header that the WBEM
3232 client populates to perform user authentication. If the WBEM client needs to perform role assumption and

3233   the WBEM server challenge is credentialrequired, the CIMRoleAuthorization header must be supplied
3234   with the appropriate credentials. The credentials supplied as part of the CIMRoleAuthorization header
3235   must use the same scheme as those specified for the authorization header, as specified in RFC2617.
3236   Therefore, both Basic and Digest authentication are possible for the role credential.

3237   If the WBEM client wishes to assume a role but does not wish to supply role credentials for server
3238   challenge credentialoptional or credentialnotrequired, the CIMRoleAuthorization header must set the
3239   auth-scheme field as specified in RFC2617 to be "role". The auth-param must contain the role name.

3240   A WBEM server that supports roles must be capable of handling the presence of credentials in the
3241   CIMRoleAuthorization header (that is auth-scheme not set to "role") regardless of whether it is expecting
3242   credentials or not. It may choose to ignore these credentials.

6.3.14 CIMStatusCodeDescription

3244   If a CIM product includes the CIMStatusCode trailer, it may also include the CIMStatusCodeDescription
3245   trailer. The value of this trailer is a string describing the nature of the error. A CIM product shall not
3246   include this trailer if the CIMStatusCode trailer is not present.

6.3.15 WBEMServerResponseTime

3248   The WBEMServerResponseTime header may be present in any CIM response message. If it is present,
3249   the header shall contain a measure, specified in microseconds, of the elapsed time required by the
3250   WBEM server to process the request and create a response. Specifically, WBEMServerResponseTime
3251   describes the time elapsed since the WBEM server received the CIM request message and the
3252   associated CIM response message was ready to send to the WBEM client.

3253   WBEMServerResponseTime = "WBEMServerResponseTime" ":", where the response time must be
3254   representable as a 64-bit unsigned integer value. If the actual elapsed time exceeds the maximum
3255   representable value, then the maximum value shall be returned. If the actual elapsed time is less than 1
3256   microsecond, then a 0 shall be returned.

3257   Although a WBEM client may ignore the WBEMServerResponseTime header, it shall allow this header to
3258   be included in a response.

7   HTTP Requirements and Usage

3260   This clause describes HTTP support and the use of standard headers.

7.1   HTTP and HTTPS Support

3262   CIM products shall support CIM-XML messages in HTTP. The following applies to this case:

3263   •   CIM products should support HTTP/1.1 as defined in RFC2616.

DEPRECATED

3265   CIM products may support HTTP/1.0 as defined in RFC1945.

3266   •   Support for HTTP/1.0 is deprecated since version 1.4 of this document; HTTP/1.1 should be
3267   supported instead.

DEPRECATED

3269   CIM products should support CIM-XML messages in HTTPS. If they do, the following applies to this case:

3270   •   CIM products shall support HTTPS as defined in RFC2818. This includes the use of HTTP
3271   within HTTPS, as defined in RFC2818.

3272   NOTE   RFC2818 describes the use of TLS 1.0 and higher but not the use of SSL 2.0 or 3.0.

3273   •   Within their support of HTTPS, CIM products:

3274   –   shall support TLS 1.0 (also known as SSL 3.1) as defined in RFC2246. Note that TLS 1.0
3275   implementations may be vulnerable when using CBC cipher suites

3276   –   should support TLS 1.1 as defined in RFC4346

3277   –   should support TLS 1.2 as defined in RFC5246

3278   –   should not support SSL 2.0 or SSL 3.0 because of known security issues in these versions

3279   NOTE   RFC5246 describes in Appendix E "Backward Compatibility" how the secure sockets layer can
3280   be negotiated.

3281 Requirements and considerations for authentication and encryption between CIM products are described
3282 in 7.4.

3283 CIM products that use extension headers as defined in this document shall conform to the requirements
3284 defined in RFC2774 for their use.

3285   7.2   Use of Standard HTTP Headers

3286 Unless otherwise stated in this document, CIM products shall comply with the requirements on the use of
3287 standard HTTP headers described in RFC1945 and RFC2616. This clause defines only *additional*
3288 requirements on CIM products with respect to the use of these standard HTTP headers in a CIM-XML
3289 message.

3290 Note that CIM products should not use HTTP headers defined in RFC2068 but deprecated in RFC2616
3291 (for example, Public, Content-Base).

3292   7.2.1   Accept

3293 If a WBEM client includes an Accept header in a request, it shall specify a value that allows the WBEM
3294 server to return an entity body of "text/xml" or "application/xml" in the response.

3295 A WBEM server or WBEM listener shall accept any value for this header stating that "text/xml" or
3296 "application/xml" is an acceptable type for a response entity. A WBEM server or WBEM listener should
3297 return "406 Not Acceptable" if the Accept header indicates that neither of these content types is
3298 acceptable.

3299 If a WBEM server or WBEM listener accepts a request to return an entity of a type other than "text/xml" or
3300 "application/xml", the nature of the response is outside the scope of this document.

3301   7.2.2   Accept-Charset

3302 If a WBEM client includes an Accept-Charset header in a request, it shall specify a value that allows the
3303 WBEM server or WBEM listener to return an entity body using the character set "UTF-8".

3304 A WBEM server or WBEM listener shall accept any value for this header asserting that "UTF-8" is an
3305 acceptable character set for a response entity. If the client does not provide an Accept-Charset, then
3306 "UTF-8" should be assumed by the WBEM server or WBEM listener.

3307   `Accept-Charset: UTF-8`

3308 A WBEM server or WBEM listener shall return "406 Not Acceptable" if the character set requested in the
3309 Accept-Charset header is not supported.

3310 If a WBEM server or WBEM listener accepts a request to return an entity using a character set other than
3311 "UTF-8", the behavior of the subsequent WBEM client and WBEM server interaction is outside the scope
3312 of this document. See 7.8 for details.

7.2.3 Accept-Encoding

3314 If a WBEM client includes an Accept-Encoding header in a request, it shall specify a q value that allows
3315 the WBEM server or WBEM listener to use the "Identity" encoding. The value shall be greater than 0 or
3316 not specified.

```
3317      Accept-Encoding: Identity
3318      Accept-Encoding: Identity; q=1.0
```

3319 A WBEM server or WBEM listener shall accept any value for this header asserting that "Identity" is an
3320 acceptable encoding for the response entity.

3321 A WBEM server or WBEM listener shall return "406 Not Acceptable" if the Accept-Encoding header
3322 indicates that the requested encoding is not acceptable.

7.2.4 Accept-Language

3324 If a WBEM client includes an Accept-Language header in a request, it shall request a language-range,
3325 special-range, or both. The WBEM client shall also allow any language to be returned if the requested
3326 languages cannot be supported. This is accomplished by including the special-range, "*". The WBEM
3327 client may request multiple languages. Each language has equal priority, unless a q value is provided.

```
3328      Accept-Language: zh, *
3329      Accept-Language: zh;q=1.0, en;q=.7, *
```

3330 Each CIM element in the response should be localized in only one language. A CIM element shall not be
3331 duplicated in the response because it is localized in more than one language.

3332 WBEM servers may support multiple languages. A CIM product shall interpret the use of the special-
3333 range value, "*", as a request to return the response content using the default language defined for the
3334 target processing the request. Multiple targets, with different default language settings, may participate in
3335 the construction of a response. (See RFC2616 section 3.10 and ISO 639-1.)

3336 See 7.8 for more information.

7.2.5 Accept-Ranges

3338 WBEM clients shall not include the Accept-Ranges header in a request. A WBEM server or WBEM
3339 listener shall reject a request that includes an Accept-Range header with a status of "406 Not
3340 Acceptable".

7.2.6 Allow

3342 If a WBEM server or WBEM listener is returning a "405 Method Not Allowed" response to a CIM-XML
3343 message request, then the Allow header shall include either M-POST or POST. Whether it includes any
3344 other HTTP methods is outside the scope of this document.

7.2.7 Authorization

3346 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.

3388 A CIM-XML message may contain a Content-Language header. The value of the Content-Language
3389 header in a CIM response message shall be consistent with the Accept-Language values specified in the
3390 corresponding CIM request message. If the WBEM server cannot determine one or more of the content
3391 languages used to construct the response, then the Content-Language entity shall not be returned.

3392 Multiple targets using different Content-Language values may participate in constructing a response. The
3393 Content-Language field shall reflect all Content-Language values used to construct the response. The
3394 content of a CIM-XML message may contain elements in languages not listed in the Content-Language
3395 field.

3396   ```
      Content-Language: en
      ```

3397 See 7.8 for details.

3398 7.2.12 Content-Range

3399 WBEM clients, WBEM servers, and WBEM listeners shall not use this header.

3400 7.2.13 Content-Type

3401 WBEM clients, WBEM servers, and WBEM listeners shall specify (and accept) a media type for the
3402 Content-Type header of either "text/xml" or "application/xml" as defined in RFC2376. In addition, they
3403 may specify and shall accept a "charset" parameter as defined in RFC2616. If a "charset" parameter is
3404 specified, it shall have the value "utf-8" either with or without surrounding double quotes. The sending
3405 side should use the form without double quotes. The receiving side shall support both forms. If a "charset"
3406 parameter is not specified, the receiving side shall assume "utf-8" as a default.

3407 Examples of valid Content-Type headers are:

3408   ```
      Content-type: text/xml
3409   Content-type: text/xml; charset=utf-8
3410   Content-type: text/xml; charset="utf-8"
3411   Content-type: application/xml
3412   Content-type: application/xml; charset=utf-8
3413   Content-type: application/xml; charset="utf-8"
      ```

### 3414  7.2.14 Expires

3415  For the reasons described in 7.2.8, a WBEM server or WBEM listener shall not include an Expires header
3416  in a CIM-XML message response that might indicate to a cache that the response can be cached.

### 3417  7.2.15 If-Range

3418  WBEM clients, WBEM servers, and WBEM listeners shall not use this header.

### 3419  7.2.16 Proxy-Authenticate

3420  See 7.4 for details.

### 3421  7.2.17 Range

3422  WBEM clients, WBEM servers, and WBEM listeners shall not use this header.

### 3423  7.2.18 WWW-Authenticate

3424  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`.

  – **(DEPRECATED)** 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

3467        The WBEM server or WBEM listener is configured to require that the proxy authenticate itself
3468        before it can issue CIM-XML message requests on behalf of a WBEM client to the WBEM
3469        server or WBEM listener.

3470    Assuming that the CIM extension headers are correct, a validating WBEM server or WBEM listener (one
3471    that enforces the validity of the CIM-XML message request with respect to the CIM XML DTD) shall use
3472    the following status code when processing the entity body containing the CIM-XML message request:

3473    • 400 Bad Request

3474        The entity body defining the CIM-XML message request is not well-formed or not valid with
3475        respect to the CIM XML DTD. The WBEM server or WBEM listener shall include a CIMError
3476        header in the response with a value of `request-not-well-formed` or `request-not-`
3477        `valid` (as appropriate).

3478    A loosely-validating WBEM server or WBEM listener only enforces the CIM-XML message request to be
3479    loosely valid. Therefore, it may reject a CIM-XML message request that is not loosely valid with an HTTP
3480    status code of 400 (Bad Request) before further processing. In this case, the WBEM server or WBEM
3481    listener shall include a CIMError header in the response with a value of `request-not-loosely-`
3482    `valid`.

3483    A loosely-validating WBEM server or WBEM listener shall reject a CIM-XML message request that is not
3484    well-formed with an HTTP status code of 400 (Bad Request). In this case, the WBEM server or WBEM
3485    listener shall include a CIMError header in the response with a value of `request-not-well-formed`.

3486    A loosely-validating WBEM server or WBEM listener shall not reject an invalid CIM-XML message request
3487    that is loosely valid in the XML sense.

3488    A loosely-validating WBEM server or WBEM listener shall ultimately signal an error to the WBEM client if
3489    the CIM-XML message request is not loosely valid. That is, the request is missing required content or the
3490    required content is incorrect, such as an attribute with an invalid value according to the CIM XML DTD. It
3491    is not mandated to reject a CIM-XML message request before processing, for to do otherwise would
3492    compel the WBEM server or WBEM listener to check the complete request before processing can begin
3493    and this would be as expensive as requiring the WBEM server or WBEM listener to fully validate the
3494    request. Therefore, a loosely-validating server or listener may elect to begin processing the request and
3495    issuing a response (with an HTTP success status code) before verifying that the entire request is loosely
3496    valid.

3497    A WBEM client may use the CIMValidation header mechanism to determine whether a WBEM server or
3498    WBEM listener is validating or loosely-validating.

3499    Assuming that the CIM-XML message request is correctly formed as previously described, the WBEM
3500    server or WBEM listener shall process the request accordingly and return a CIM-XML message response.

3501    The entity body shall be a correct CIM-XML message response for that request.

3502    If the CIM-XML message response contains an entity that is a simple message response, then the
3503    response status shall be "200 OK". Otherwise, the response status shall be "207 Multistatus".

3504    ## 7.4   Security Considerations

3505    This subclause describes requirements and considerations for authentication and message encryption
3506    between CIM products.

3507    ### 7.4.1   Authentication

3508    This subclause describes requirements and considerations for authentication between CIM products.
3509    Specifically, authentication happens from WBEM clients to WBEM servers for CIM-XML operation

3510   messages, and from WBEM servers to WBEM listeners for CIM-XML export messages. The
3511   authentication mechanisms defined in this subclause apply to both HTTP and HTTPS.

3512   CIM products may support operating without the use of authentication. This practice is not recommended
3513   and should only be done in environments where lack of network privacy is not an issue (for example, in a
3514   physically secure private network or on the same operating system).

3515   Basic authentication is described in RFC1945 and RFC2068. Digest authentication is defined in
3516   RFC2069. Both authentication schemes are covered in a consolidated document (RFC2617), which also
3517   makes a number of improvements to the original specification of digest authentication. This document
3518   requires conformance to RFC2617 but not to the earlier documents.

3519   Basic authentication provides a very rudimentary level of authentication, with the major weakness that the
3520   client password is sent over the wire in unencrypted form (unless HTTPS is used)..

3521   CIM products may support basic authentication as defined in RFC2617. Basic authentication without
3522   HTTPS should only be used in environments where lack of network privacy is not an issue.

3523   Digest authentication verifies that both parties share a common secret without having to send that secret.

3524   CIM products should support digest authentication as defined in RFC2617.

3525   CIM products may support authentication mechanisms not covered by RFC2617. One example are public
3526   key certificates as defined in X.509.

3527   WBEM servers and WBEM listeners should require that WBEM clients and WBEM servers, respectively,
3528   authenticate themselves. This document does not mandate this because it is recognized that in some
3529   circumstances the WBEM server or WBEM listener may not require or wish the overhead of employing
3530   authentication. WBEM servers and WBEM listeners should carefully consider the performance/security
3531   tradeoffs in determining how often to issue challenges to WBEM clients and WBEM servers, respectively.

3532   A WBEM server or WBEM listener that returns a "401 Unauthorized" response to a CIM message request
3533   shall include one WWW-Authenticate response-header indicating one supported authentication
3534   mechanism. This document does not mandate use of basic or digest authentication because it is
3535   recognized that in some circumstances the WBEM server or WBEM listener may use bespoke
3536   authentication mechanisms not covered by RFC2617. Similar considerations apply to the use of the
3537   Proxy-Authenticate response-header in "407 Proxy Authentication Required".

3538   ### 7.4.2   Message Encryption

3539   Encryption of messages between CIM products is supported by the use of HTTPS in the communication
3540   between CIM products. Requirements for the use of HTTPS and its underlying secure sockets are
3541   defined in 7.1.

3542   The following requirements on cipher suites apply to CIM products that support HTTPS:

3543   • The TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value 0x0013)
3544     shall be supported when using TLS 1.0. Note that RFC2246 defines this cipher suite to be
3545     mandatory for TLS 1.0

3546   • The TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value 0x000A) shall
3547     be supported when using TLS 1.1. Note that RFC4346 defines this cipher suite to be mandatory
3548     for TLS 1.1

3549   • The TLS_RSA_WITH_AES_128_CBC_SHA cipher suite (hexadecimal value 0x002F) shall be
3550     supported when using TLS 1.2. Note that RFC5246 defines this cipher suite to be mandatory for
3551     TLS 1.2

3552        • The TLS_RSA_WITH_AES_128_CBC_SHA256 cipher suite (hexadecimal value 0x003C)
3553           should be supported when using TLS 1.2, in order to meet the transition to a security strength of
3554           112 bits (guidance is provided in NIST 800-57 and NIST 800-131A)

3555        • Any additional cipher suites may be supported

## 7.5   Determining WBEM server Capabilities

3557   If a WBEM server can return capabilities information, there are two techniques for returning this
3558   information as defined in this document:

3559        • The preferred technique is through the use of the classes defined in 7.5.1.

3560        • Alternatively, use of the HTTP OPTIONS method as defined in 7.5.2 is allowed because
3561           historically it is the original technique defined for requesting capabilities information.

3562   Use of the CIM classes defined in 7.5.1 is strongly encouraged and it is expected that this method will be
3563   enhanced and extended in the future to provide more capabilities information. The future use of the HTTP
3564   OPTIONS method to determine capabilities of WBEM servers is discouraged. It will probably not be
3565   expanded significantly and may be reviewed for possible deprecation in the next major revision of this
3566   document.

### 7.5.1   Determining WBEM server Capabilities through CIM Classes (DEPRECATED)

3568   **DEPRECATION NOTE:  This section was deprecated in version 1.4 of this document because it**
3569   **was determined that this is outside the scope of this document. The WBEM Server Profile and**
3570   **SNIA CIM Server Profile contain the same type of information.**

3571   A set of CIM classes is defined specifically to return WBEM server capabilities information as follows:

3572        • CIM_ObjectManager

3573           This class is a type of CIM_Service that defines the capabilities of the target WBEM server.

3574        • CIM_ObjectManagerCommunicationMechanism

3575           This class describes access to the target WBEM server. It defines the capabilities of the WBEM
3576           server that are available through the target Object Manager Communication mechanism. A
3577           WBEM server is allowed to support different capabilities through different communication
3578           mechanisms.

3579        • CIM_CIMXMLCommunicationMechanism

3580           This class specializes on ObjectManagerCommunicationMechanism, adding properties specific
3581           to the CIM-XML encoding and protocol.

3582        • CIM_CommMechanismForManager

3583           This association between CIM_ObjectManager and
3584           CIM_ObjectManagerCommunicationMechanism defines the communications protocols (and
3585           corresponding capabilities) available on the target WBEM server through the
3586           ObjectManagerCommunicationMechanism instances.

3587   A WBEM client may use instances of these CIM classes to determine the CIM capabilities (if any) of the
3588   target WBEM server. A WBEM server that supports capabilities determination through these classes shall
3589   support at least the Enumerate Instance and Get Instance operations for the classes. The use of other
3590   methods of the basic read profile is optional. A WBEM server that does not support the determination of
3591   CIM capabilities through these classes shall return CIM_ERR_NOT_FOUND to any instance or class
3592   request on these classes. These classes shall not be used for reporting any other information than
3593   capabilities of the target WBEM server.

3594    To provide interoperability, the CIM object manager classes shall exist in a well-known namespace.
3595    Because there is no discovery mechanism that can define this well-known namespace to a WBEM client,
3596    it shall be one or more predefined namespaces. Therefore, to ensure interoperability, we recommend that
3597    pending future extensions of the WBEM specifications include discovery tools that define a namespace
3598    for these classes in a WBEM server; these predefined namespaces should exist in either the root
3599    namespace or in the /root/CIMV2 namespace.

3600    A WBEM server that supports capabilities reporting through these classes shall correctly report the
3601    current actual capabilities of the target WBEM server and shall report on all capabilities defined. A WBEM
3602    server is allowed to report "none" if the capability does not exist or "unknown" if the status of the capability
3603    is unknown at the time of the request for those properties where these choices exist in the properties
3604    definition. Because the CIM_ObjectManager object provides information on the target WBEM server, only
3605    a single instance of this class may exist in a WBEM server.

3606    The capabilities to be reported through the CIM_ObjectManagerCommunicationMechanism are as
3607    follows:

3608       •   CommunicationMechanism property, which defines the communication protocol for the
3609          CommunicationMechanism object. A compliant WBEM server shall include the CIM-XML
3610          protocol for at least one ObjectManagerCommunicationMechanism instance.

3611       •   ProfilesSupported property, which defines the functional profiles supported as defined in clause
3612          5.4.4. All WBEM servers shall support the basic-read functional group. All WBEM clients may
3613          assume that any WBEM server supports the basic-read functional group. The list of functional
3614          groups returned by a WBEM server shall contain the basic-read group and shall not contain
3615          duplicates. WBEM clients shall ignore duplicate entries in the functional-group list. If a functional
3616          group is included in the list, the WBEM client shall assume that all other groups on which it
3617          depends (according to the rules defined in 5.4.4) are also supported. A WBEM server should
3618          not explicitly include a functional group in the list whose presence may be inferred implicitly by a
3619          dependency. Support for a functional group does not imply that any method from that group will
3620          always succeed. Rather, the absence of the functional group from this list (whether explicit or
3621          implied) indicates to the WBEM client that methods in that group will never succeed.

3622       •   MultipleOperationsSupported property, which defines whether the target WBEM server supports
3623          multiple operation requests as defined in 5.4.2. `True` in this property indicates that the WBEM
3624          server can accept and process multiple operation requests. `False` indicates that the WBEM
3625          server can accept only single operation requests.

3626       •   AuthenticationMechanismsSupported property, which defines the authentication mechanisms
3627          supported by the target WBEM server as defined in 7.4.

3628       •   PulledEnumerationClosureOnExceedingServerLimits property, which indicates whether the
3629          WBEM server supports closure of Pulled Enumeration sessions based upon exceeding server
3630          limits.

3631       •   PulledEnumerationContinuationOnErrorSupported property, which indicates whether the WBEM
3632          server supports continuation on error for Pulled enumerations.

3633       •   PulledEnumerationMinimumOperationTimeout (PulledEnumerationMaximumOperationTimeout)
3634          property, which indicates the minimum (maximum) operation timeout allowed by the WBEM
3635          server for Pulled enumerations.

3636    Compliant WBEM servers may report additional capabilities for the CommunicationMechanism Functional
3637    Profiles, QueryLanguageSupported, and AuthenticationMechanismSupported by defining the "other"
3638    enumeration in the property and returning additional information in the associated "additional capabilities"
3639    property.

3640 ## 7.5.2    Determining WBEM server Capabilities through the HTTP Options

3641 A WBEM client may use the OPTIONS method to determine the CIM capabilities (if any) of the target
3642 server. A WBEM server may support the OPTIONS method (for example, WBEM servers supporting only
3643 HTTP/1.0 would not support OPTIONS).

3644 To support the ability for a WBEM server to declare its CIM capabilities independently of HTTP, the DMTF
3645 intends to publish a CIM schema (in a separate document) describing such capabilities. In particular, this
3646 mechanism would allow servers that do not support the OPTIONS method to declare their capabilities to
3647 a client.

3648 If a WBEM server supports the OPTIONS method, it should return the following headers in the response:

3649 • CIM Extension Header CIMProtocolVersion, which provides a way for a client to discover the
3650 version of the CIM HTTP mapping supported by the WBEM server.

3651 • **(DEPRECATED)** CIM Extension Header CIMSupportedFunctionalGroups, which provides a
3652 way for a client to discover the CIM operations supported by the WBEM server.

3653 • **(DEPRECATED)** CIM Extension Header CIMSupportsMultipleOperations, which provides a way
3654 for the client to discover whether the WBEM server can support Multiple Operation Requests.

3655 In addition, if the WBEM server supports one or more query languages for the ExecQuery operation (see
3656 5.4.2.13), it should return the following header in the response:

3657 • CIM Extension Header CIMSupportedQueryLanguages, which allows the client to discover the
3658 query languages supported by the WBEM server for the ExecQuery operation.

3659 In addition, if the WBEM server runs in a fixed validation mode, it should return the following header in the
3660 response:

3661 • CIM Extension Header CIMValidation, which allows the client to determine whether the WBEM
3662 server is strictly validating or loosely validating.

3663 If the CIMProtocolVersion, CIMSupportedFunctionalGroups**(DEPRECATED)**,
3664 CIMSupportsMultipleOperations**(DEPRECATED)**, CIMValidation, or CIMSupportedQueryLanguages
3665 extension headers are included in the response, the WBEM server shall declare them as optional
3666 extension headers using the "Opt" header defined in RFC2774.

3667 The full format of the "Opt" header declaration for this document is:

```
3668    Opt               = "Opt" ":" "http://www.dmtf.org/cim/mapping/http/v1.0"
3669                          ";" "ns" "=" header-prefix
3670
3671    header-prefix   = 2*DIGIT
```

3672 This header-prefix should be generated at random on a per-HTTP message basis and should not
3673 necessarily be a specific number.

3674 EXAMPLE: The following is a fragment of a legitimate OPTIONS response from a WBEM server:

```
3675    HTTP/1.1 200 OK
3676    Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=77
3677    77-CIMProtocolVersion: 1.0
3678    77-CIMSupportedFunctionalGroups: basic-read
3679    77-CIMBatch
3680    77-CIMSupportedQueryLanguages: wql
3681    ...
```

3682 **7.5.2.1 CIMSupportedFunctionalGroups** (DEPRECATED)

3683 **DEPRECATION NOTE: This section was deprecated in version 1.4 of this document and there is**
3684 **no replacement.**

3685 The CIMSupportedFunctionalGroups extension header should be returned by a WBEM server in any
3686 OPTIONS response. It shall not be returned in any other scenario.

3687 This header is defined as follows:

```
3688        CIMSupportedFunctionalGroups = "CIMSupportedFunctionalGroups" ":"
3689                                    1#functional-group
3690
3691        functional-group = "basic-read" |
3692           "basic-write" |
3693           "schema-manipulation" |
3694           "instance-manipulation" |
3695           "qualifier-declaration" |
3696           "association-traversal" |
3697           "query-execution"
```

3698 The functional group definitions correspond directly to those listed in 5.5.3. All WBEM servers shall
3699 support the basic-read functional group. All WBEM clients may assume that any WBEM server supports
3700 the basic-read functional group.

3701 The list of functional groups returned by a WBEM server shall contain the basic-read group and shall not
3702 contain any duplicates. WBEM clients shall ignore any duplicate entries in the functional-group list.

3703 If a functional group is included in the list, the WBEM client shall assume that all other groups on which it
3704 depends (according to the rules defined in 5.5.3) are also supported. A WBEM server should not explicitly
3705 include a functional group in the list if the presence of the group may be implied by a dependency.

3706 EXAMPLE: The following HTTP response message indicates that the WBEM server supports instance-
3707 manipulation, association-traversal, basic-write, and basic-read.

```
3708        HTTP/1.1 200 OK
3709        Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=77
3710        77-CIMProtocolVersion: 1.0
3711        77-CIMSupportedFunctionalGroups: association-traversal, instance-manipulation
3712        ...
```

3713 Support for a functional group does *not* imply that any method from that group will always succeed.
3714 Rather, the absence (whether explicit or implied) of the functional group from this header is an indication
3715 to the WBEM client that methods in that group will *never* succeed.

3716 **7.5.2.2 CIMSupportsMultipleOperations** (DEPRECATED)

3717 **DEPRECATION NOTE: This section was deprecated in version 1.4 of this document and there is**
3718 **no replacement.**

3719 The CIMSupportsMultipleOperations extension header shall be returned in an OPTIONS response by any
3720 WBEM server that supports Multiple Operation Requests. It shall not be returned in any other
3721 circumstances.

3722 This header is defined as follows:

```
3723        CIMSupportsMultipleOperations  = "CIMSupportsMultipleOperations"
```

3724 The presence of this header indicates that the WBEM server can accept and process multiple operation
3725 requests. The absence of this header indicates that the WBEM server can only accept and process
3726 Simple Operation Requests.

7.5.2.3 **CIMSupportedQueryLanguages** (DEPRECATED)

3728 The CIMSupportedQueryLanguages extension header identifies the query languages supported by the
3729 WBEM server for the ExecQuery operation (see 5.4.2.13).

3730 **DEPRECATION NOTE:** The CIMSupportedQueryLanguages extension header has been deprecated in
3731 version 1.4 of this document, because it was used only for the ExecQuery operation.

3732 The CIMSupportedQueryLanguages extension header should be returned in any OPTIONS response by
3733 a WBEM server that supports at least one such query language. It shall not be returned in any other
3734 scenario.

3735 This header is defined as follows (`token` has the meaning conferred by RFC1945 and RFC2616):

3736 ```
     CIMSupportedQueryLanguages = "CIMSupportedQueryLanguages" ":" 1#query-language
```
3737
3738 ```
     query-language     = token
```

3739 The `query-language` value shall be treated as case-insensitive. It is anticipated that query languages
3740 will be submitted for approval to the DMTF, and each submission will define a value for this token to
3741 enable it to be specified in this header.

7.5.2.4 **CIMValidation**

3743 The CIMValidation extension header may be returned by a WBEM server to provide information about the
3744 level of validation of CIM-XML operation request messages.

3745 This header is defined as follows:

3746 ```
     CIMValidation = "CIMValidation" ":" validation-level
```
3747
3748 ```
     validation-level = "validating" | "loosely-validating"
```

3749 A validation-level of `validating` indicates that the WBEM server always applies strict validation of each
3750 CIM-XML operation request. A validation-level of `loosely-validating` indicates that the WBEM
3751 server applies loose validation of each CIM-XML operation request.

3752 In the absence of this header, a WBEM client should assume that the WBEM server operates in strict
3753 validation mode.

7.6 Other HTTP Methods

3755 This document does not in any way define or constrain the way a WBEM client, WBEM server, or WBEM
3756 listener uses any HTTP method other than those explicitly cited.

7.7 Discovery and Addressing

3758 The target URI of the CIM-XML operation request is defined as the location of the WBEM server. This
3759 document does not constrain the format of this URI other than it should be a valid URI (RFC2396) for
3760 describing an HTTP-addressable resource.

3761 An HTTP server that supports the CIM mapping defined in this document, and which supports the
3762 OPTIONS method, should include the following CIM extension header in an OPTIONS response:

3763   •   CIMOM

3764   This header is defined as follows:

3765   `CIMOM       = "CIMOM" ":" (absoluteURI | relativeURI)`

3766   The terms `absoluteURI` and `relativeURI` are taken from RFC2616; they indicate the location of the
3767   WBEM server for this HTTP server.

3768   If the CIMOM extension header is included in the response, the WBEM server shall declare it an optional
3769   extension header as described in 7.5.

3770   A WBEM client that needs to communicate with a WBEM server on an HTTP server should try an
3771   OPTIONS request to that HTTP server. If the OPTIONS request fails or the response does not include the
3772   CIM-CIMOM extension header, the WBEM client may assume that the value of CIM-CIMOM is the
3773   relative URI `cimom`.

3774   The DMTF recommends the use of the following well-known IP ports in compliant WBEM servers. This is
3775   a recommendation and not a requirement. The DMTF has registered these port addresses with IANA, so
3776   they are for the exclusive use of the DMTF.

3777   •   CIM-XML (HTTP)   5988/tcp

3778   •   CIM-XML (HTTP)   5988/udp

3779   •   CIM-XML (HTTPS)   5989/tcp

3780   •   CIM-XML (HTTPS)   5989/udp

3781   Other discovery mechanisms are outside the scope of this version of the specification.

3782   EXAMPLE 1:

3783   This example shows an HTTP server located at http://www.dmtf.org/ issuing an OPTIONS response
3784   to an HTTP client to indicate that its WBEM server is located at http://www.dmtf.org/access/cimom.

3785   `HTTP/1.1 200 OK`
3786   `Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=48`
3787   `48-CIMOM: /access/cimom`
3788   `...`

3789   EXAMPLE 2:

3790   If an HTTP server located at http://www.dmtf.org/ responds with a "501 Not Implemented" to an
3791   OPTIONS request from a WBEM client, the WBEM client may then try to contact the WBEM server
3792   at http://www.dmtf.org/cimom.

3793   7.8   Internationalization Considerations

3794   This clause defines the capabilities of the CIM HTTP mapping with respect to IETF policy guidelines on
3795   character sets and languages (RFC2277).

3796   In this document, human-readable fields are contained within a response or request entity body. In all
3797   cases, a human-readable content is encoded using XML (which explicitly provides for character set
3798   tagging and encoding) and requires that XML processors read XML elements encoded, at minimum,
3799   using the UTF-8 (RFC2279) encoding of the ISO 10646 multilingual plane.

3800   Properties that are not of type string or string array shall not be localized.

3801   Because keys are writeable only on instantiation, key values shall not be localized. See DSP0004 for
3802   details.

3803 XML examples in this document demonstrate the use of the charset parameter of the Content-Type
3804 header, as defined in RFC2616, as well as the XML attribute on the <?xml> processing instruction, which
3805 together provide charset identification information for MIME and XML processors. This document
3806 mandates that conforming applications shall support at least the "UTF-8" charset encoding (RFC2277) in
3807 the Content-Type header and shall support the "UTF-8" value for the XML `encoding` attribute.

3808 XML also provides a language tagging capability for specifying the language of the contents of a
3809 particular XML element, based on use of IANA registered language tags (RFC1766) in combination with
3810 ISO 639-1, in the xml:lang attribute of an XML element to identify the language of its content and
3811 attributes. Section 3.10 of RFC2616 defines how the two-character ISO 639-1 language code is used as
3812 the primary-tag. The language-tag shall be registered by IANA.

3813 DSP0201 declares this attribute on any XML elements. Therefore, conforming applications should use
3814 this attribute when specifying the language in which a particular element is encoded for string and string
3815 array attributes and qualifiers. See the usage rules on this element, which are defined by the World Wide
3816 Web Consortium in *XML 1.0, second edition*. The attribute may be scoped by the instance or a class and
3817 should not be scoped by a property because instances or classes should be localized in one language.

3818 This document defines several names of HTTP headers and their values. These names are constructed
3819 using standard encoding practices so that they always have an HTTP-safe ASCII representation.
3820 Because these headers are not usually visible to users, they do not need to support encoding in multiple
3821 character sets.

3822 DSP0201 introduces several XML element names. Similarly, these names are not visible to an end user
3823 and do not need to support multiple character set encodings.

3824 The CIM model (DSP0004) defines the subset of the Unicode character set that can be used to name
3825 CIM elements (classes, instances, methods, properties, qualifiers, and method parameters). In general,
3826 these characters appear as the value of XML attributes or as element content and are not displayed to
3827 end users.

3828 Negotiation and notification of language settings is effected in this mapping using the standard Accept-
3829 Language and Content-Language headers defined in RFC1945 and RFC2616.

ANNEX A
(Informative)

3832

3833

3834 # Examples of Message Exchanges

3835 This annex illustrates the protocol defined in this document with examples of valid HTTP
3836 request/response exchanges. The examples are for illustration purposes only and are not considered part
3837 of the specification.

3838 For clarity, additional white space is included in the examples, but such white space is not an intrinsic part
3839 of such XML documents.

A.1   Retrieval of a Single Class Definition
3840

3841 The following HTTP request illustrates how a client requests the class CIM_VideoBIOSElement.

```
3842    M-POST /cimom HTTP/1.1
3843    HOST: http://www.myhost.com/
3844    Content-Type: application/xml; charset=utf-8
3845    Content-Length: xxxx
3846    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3847    73-CIMOperation: MethodCall
3848    73-CIMMethod: GetClass
3849    73-CIMObject: root/cimv2
3850
3851    <?xml version="1.0" encoding="utf-8" ?>
3852    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3853      <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3854        <SIMPLEREQ>
3855          <IMETHODCALL NAME="GetClass">
3856            <LOCALNAMESPACEPATH>
3857              <NAMESPACE NAME="root"/>
3858              <NAMESPACE NAME="cimv2"/>
3859            </LOCALNAMESPACEPATH>
3860            <IPARAMVALUE NAME="ClassName">
3861              <CLASSNAME NAME="CIM_VideoBIOSElement"/>
3862            </IPARAMVALUE>
3863            <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
3864          </IMETHODCALL>
3865        </SIMPLEREQ>
3866      </MESSAGE>
3867    </CIM>
```

3868 Following is an HTTP response to the preceding request indicating success of the requested operation.
3869 For clarity of exposition, the complete definition of the returned <CLASS> element is not shown.

```
3870    HTTP/1.1 200 OK
3871    Content-Type: application/xml; charset=utf-8
3872    Content-Length: xxxx
3873    Ext:
3874    Cache-Control: no-cache
3875    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
```

```
3876        73-CIMOperation: MethodResponse
3877
3878        <?xml version="1.0" encoding="utf-8" ?>
3879        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3880          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3881            <SIMPLERSP>
3882              <IMETHODRESPONSE NAME="GetClass">
3883                <IRETURNVALUE>
3884                  <CLASS NAME="CIM_VideoBIOSElement"
3885                         SUPERCLASS="CIM_SoftwareElement">
3886                    ...
3887                  </CLASS>
3888                </IRETURNVALUE>
3889              </IMETHODRESPONSE>
3890            </SIMPLERSP>
3891          </MESSAGE>
3892        </CIM>
```

3893   A.2   Retrieval of a Single Instance Definition

3894   The following HTTP request illustrates how a client requests the instance MyClass.MyKey="S3".

```
3895        M-POST /cimom HTTP/1.1
3896        HOST: http://www.myhost.com/
3897        Content-Type: application/xml; charset=utf-8
3898        Content-Length: xxxx
3899        Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3900        73-CIMOperation: MethodCall
3901        73-CIMMethod: GetInstance
3902        73-CIMObject: root%2FmyNamespace
3903
3904        <?xml version="1.0" encoding="utf-8" ?>
3905        <CIM CIMVERSION="2.0" DTDVERSION="1.1">
3906          <MESSAGE ID="87855" PROTOCOLVERSION="1.0">
3907            <SIMPLEREQ>
3908              <IMETHODCALL NAME="GetInstance">
3909                <LOCALNAMESPACEPATH>
3910                  <NAMESPACE NAME="root"/>
3911                  <NAMESPACE NAME="myNamespace"/>
3912                </LOCALNAMESPACEPATH>
3913                <IPARAMVALUE NAME="InstanceName">
3914                  <INSTANCENAME CLASSNAME="MyClass">
3915                    <KEYBINDING NAME="MyKey"><KEYVALUE>S3</KEYVALUE></KEYBINDING>
3916                  </INSTANCENAME>
3917                </IPARAMVALUE>
3918                <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
3919              </IMETHODCALL>
3920            </SIMPLEREQ>
3921          </MESSAGE>
3922        </CIM>
```

3923   Following is an HTTP response to the preceding request indicating an error because the specified
3924   instance is not found.

```
3925    HTTP/1.1 200 OK
3926    Content-Type: application/xml; charset=utf-8
3927    Content-Length: xxxx
3928    Ext:
3929    Cache-Control: no-cache
3930    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3931    73-CIMOperation: MethodResponse
3932
3933    <?xml version="1.0" encoding="utf-8" ?>
3934    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3935       <MESSAGE ID="87885" PROTOCOLVERSION="1.0">
3936          <SIMPLERSP>
3937             <IMETHODRESPONSE NAME="GetInstance">
3938                <ERROR CODE="6" DESCRIPTION="Instance of MyClass not found"/>
3939             </IMETHODRESPONSE>
3940          </SIMPLERSP>
3941       </MESSAGE>
3942    </CIM>
```

3943 A.3   Deletion of a Single Class Definition

3944 The following HTTP request illustrates how a client deletes the class CIM_VideoBIOSElement.

```
3945    M-POST /cimom HTTP/1.1
3946    HOST: http://www.myhost.com/
3947    Content-Type: application/xml; charset=utf-8
3948    Content-Length: xxxx
3949    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3950    73-CIMOperation: MethodCall
3951    73-CIMMethod: DeleteClass
3952    73-CIMObject: root/cimv2
3953
3954    <?xml version="1.0" encoding="utf-8" ?>
3955    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3956       <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3957          <SIMPLEREQ>
3958             <IMETHODCALL NAME="DeleteClass">
3959                <LOCALNAMESPACEPATH>
3960                   <NAMESPACE NAME="root"/>
3961                   <NAMESPACE NAME="cimv2"/>
3962                </LOCALNAMESPACEPATH>
3963                <IPARAMVALUE NAME="ClassName">
3964                   <CLASSNAME NAME="CIM_VideoBIOSElement"/>
3965                </IPARAMVALUE>
3966             </IMETHODCALL>
3967          </SIMPLEREQ>
3968       </MESSAGE>
3969    </CIM>
```

3970 Following is an HTTP response to the preceding request indicating failure of the preceding operation due
3971 to the inability to delete instances of the class.

```
3972    HTTP/1.1 200 OK
3973    Content-Type: application/xml; charset=utf-8
```

```
3974      Content-Length: xxxx
3975      Ext:
3976      Cache-Control: no-cache
3977      Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3978      73-CIMOperation: MethodResponse
3979
3980      <?xml version="1.0" encoding="utf-8" ?>
3981      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3982        <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3983          <SIMPLERSP>
3984            <IMETHODRESPONSE NAME="DeleteClass">
3985              <ERROR CODE="9" DESCRIPTION="Class has non-deletable instances"/>
3986            </IMETHODRESPONSE>
3987          </SIMPLERSP>
3988        </MESSAGE>
3989      </CIM>
```

A.4 Deletion of a Single Instance Definition

3990

3991 The following HTTP request illustrates how a client deletes the instance MyClass.MyKey="S3".

```
3992      M-POST /cimom HTTP/1.1
3993      HOST: http://www.myhost.com/
3994      Content-Type: application/xml; charset=utf-8
3995      Content-Length: xxxx
3996      Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3997      73-CIMOperation: MethodCall
3998      73-CIMMethod: DeleteInstance
3999      73-CIMObject: root%2FmyNamespace
4000
4001      <?xml version="1.0" encoding="utf-8" ?>
4002      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4003        <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4004          <SIMPLEREQ>
4005            <IMETHODCALL NAME="DeleteInstance">
4006              <LOCALNAMESPACEPATH>
4007                <NAMESPACE NAME="root"/>
4008                <NAMESPACE NAME="myNamespace"/>
4009              </LOCALNAMESPACEPATH>
4010              <IPARAMVALUE NAME="InstanceName">
4011                <INSTANCENAME CLASSNAME="MyClass">
4012                  <KEYBINDING NAME="MyKey">
4013                    <KEYVALUE>S3</KEYVALUE>
4014                  </KEYBINDING>
4015                </INSTANCENAME>
4016              </IPARAMVALUE>
4017            </IMETHODCALL>
4018          </SIMPLEREQ>
4019        </MESSAGE>
4020      </CIM>
```

4021 Following is an HTTP response to the preceding request indicating success of the preceding operation.

```
4022      HTTP/1.1 200 OK
```

```
4023        Content-Type: application/xml; charset=utf-8
4024        Content-Length: xxxx
4025        Ext:
4026        Cache-Control: no-cache
4027        Man: http://www.dmtf.org/cim/operation ; ns=73
4028        73-CIMOperation: MethodResponse
4029
4030        <?xml version="1.0" encoding="utf-8" ?>
4031        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4032          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4033             <SIMPLERSP>
4034                <IMETHODRESPONSE NAME="DeleteInstance"/>
4035             </SIMPLERSP>
4036          </MESSAGE>
4037        </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.

```
4042        M-POST /cimom HTTP/1.1
4043        HOST: http://www.myhost.com/
4044        Content-Type: application/xml; charset=utf-8
4045        Content-Length: xxxx
4046        Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4047        73-CIMOperation: MethodCall
4048        73-CIMMethod: CreateClass
4049        73-CIMObject: root/cimv2
4050
4051        <?xml version="1.0" encoding="utf-8" ?>
4052        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4053          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4054             <SIMPLEREQ>
4055                <IMETHODCALL NAME="CreateClass">
4056                   <LOCALNAMESPACEPATH>
4057                      <NAMESPACE NAME="root"/>
4058                      <NAMESPACE NAME="cimv2"/>
4059                   </LOCALNAMESPACEPATH>
4060                   <IPARAMVALUE NAME="NewClass">
4061                      <CLASS NAME="MySchema_VideoBIOSElement"
4062                             SUPERCLASS="CIM_VideoBIOSElement">
4063                         ...
4064                      </CLASS>
4065                   </IPARAMVALUE>
4066                </IMETHODCALL>
4067             </SIMPLEREQ>
4068          </MESSAGE>
4069        </CIM>
```

Following is an HTTP response to the preceding request indicating success of the preceding operation.

```
4071        HTTP/1.1 200 OK
```

```
4072      Content-Type: application/xml; charset=utf-8
4073      Content-Length: xxxx
4074      Ext:
4075      Cache-Control: no-cache
4076      Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4077      73-CIMOperation: MethodResponse
4078
4079      <?xml version="1.0" encoding="utf-8" ?>
4080      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4081        <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4082           <SIMPLERSP>
4083              <IMETHODRESPONSE NAME="CreateClass"/>
4084           </SIMPLERSP>
4085        </MESSAGE>
4086      </CIM>
```

4087   A.6   Creation of a Single Instance Definition

4088   The following HTTP request illustrates how a client creates an instance of the class
4089   MySchema_VideoBIOSElement. For clarity of exposition, most of the submitted <INSTANCE> element is
4090   omitted from the example.

```
4091      M-POST /cimom HTTP/1.1
4092      HOST: http://www.myhost.com/
4093      Content-Type: application/xml; charset=utf-8
4094      Content-Length: xxxx
4095      Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4096      73-CIMOperation: MethodCall
4097      73-CIMMethod: CreateInstance
4098      73-CIMObject: root/cimv2
4099
4100      <?xml version="1.0" encoding="utf-8" ?>
4101      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4102        <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4103           <SIMPLEREQ>
4104              <IMETHODCALL NAME="CreateInstance">
4105                 <LOCALNAMESPACEPATH>
4106                    <NAMESPACE NAME="root"/>
4107                    <NAMESPACE NAME="cimv2"/>
4108                 </LOCALNAMESPACEPATH>
4109                 <IPARAMVALUE NAME="NewInstance">
4110                    <INSTANCE CLASSNAME="CIM_VideoBIOSElement">
4111                        ...
4112                    </INSTANCE>
4113                 </IPARAMVALUE>
4114              </IMETHODCALL>
4115           </SIMPLEREQ>
4116        </MESSAGE>
4117      </CIM>
```

4118   Following is an HTTP response to the preceding request indicating the success of the preceding
4119   operation.

```
4120      HTTP/1.1 200 OK
```

```
4121    Content-Type: application/xml; charset=utf-8
4122    Content-Length: xxxx
4123    Ext:
4124    Cache-Control: no-cache
4125    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4126    73-CIMOperation: MethodResponse
4127
4128    <?xml version="1.0" encoding="utf-8" ?>
4129    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4130       <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4131          <SIMPLERSP>
4132             <IMETHODRESPONSE NAME="CreateInstance">
4133                <IRETURNVALUE>
4134                   <INSTANCENAME CLASSNAME="MySchema_VideoBIOSElement">
4135                      <KEYBINDING NAME="Name"><KEYVALUE>S4</KEYVALUE></KEYBINDING>
4136                   </INSTANCENAME>
4137                </IRETURNVALUE>
4138             </IRETURNVALUE>
4139          </SIMPLERSP>
4140       </MESSAGE>
4141    </CIM>
```

4142 A.7   Enumeration of Class Names

4143 The following HTTP request illustrates how a client enumerates the names of all subclasses of the class
4144 CIM_SoftwareElement.

```
4145    M-POST /cimom HTTP/1.1
4146    HOST: http://www.myhost.com/
4147    Content-Type: application/xml; charset=utf-8
4148    Content-Length: xxxx
4149    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4150    73-CIMOperation: MethodCall
4151    73-CIMMethod: EnumerateClassNames
4152    73-CIMObject: root/cimv2
4153
4154    <?xml version="1.0" encoding="utf-8" ?>
4155    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4156       <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4157          <SIMPLEREQ>
4158             <IMETHODCALL NAME="EnumerateClassNames">
4159                <LOCALNAMESPACEPATH>
4160                   <NAMESPACE NAME="root"/>
4161                   <NAMESPACE NAME="cimv2"/>
4162                </LOCALNAMESPACEPATH>
4163                <IPARAMVALUE NAME="ClassName">
4164                   <CLASSNAME NAME="CIM_SoftwareElement"/>
4165                </IPARAMVALUE>
4166                <IPARAMVALUE NAME="DeepInheritance">
4167                   <VALUE>FALSE</VALUE>
4168                </IPARAMVALUE>
4169             </IMETHODCALL>
4170          </SIMPLEREQ>
```

```
4171            </MESSAGE>
4172        </CIM>
```

4173   Following is an HTTP response to the preceding request indicating the success of the preceding
4174   operation and returning the names of the requested subclasses.

```
4175        HTTP/1.1 200 OK
4176        Content-Type: application/xml; charset=utf-8
4177        Content-Length: xxxx
4178        Ext:
4179        Cache-Control: no-cache
4180        Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4181        73-CIMOperation: MethodResponse
4182
4183        <?xml version="1.0" encoding="utf-8" ?>
4184        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4185            <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4186                <SIMPLERSP>
4187                    <IMETHODRESPONSE NAME="EnumerateClassNames">
4188                        <IRETURNVALUE>
4189                            <CLASSNAME NAME="CIM_BIOSElement"/>
4190                            <CLASSNAME NAME="CIM_VideoBOISElement"/>
4191                        </IRETURNVALUE>
4192                    </IMETHODRESPONSE>
4193                </SIMPLERSP>
4194            </MESSAGE>
4195        </CIM>
```

4196   A.8   Enumeration of Instances

4197   The following HTTP request illustrates how a client enumerates all instances of the class
4198   CIM_LogicalDisk. For clarity of exposition, most of the returned instances are omitted from the example.

```
4199        M-POST /cimom HTTP/1.1
4200        HOST: http://www.myhost.com/
4201        Content-Type: application/xml; charset=utf-8
4202        Content-Length: xxxx
4203        Man: http://www.dmtf.org/cim/operation ; ns=73
4204        73-CIMOperation: MethodCall
4205        73-CIMMethod: EnumerateInstances
4206        73-CIMObject: root/cimv2
4207
4208        <?xml version="1.0" encoding="utf-8" ?>
4209        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4210            <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4211                <SIMPLEREQ>
4212                    <IMETHODCALL NAME="EnumerateInstances">
4213                        <LOCALNAMESPACEPATH>
4214                            <NAMESPACE NAME="root"/>
4215                            <NAMESPACE NAME="cimv2"/>
4216                        </LOCALNAMESPACEPATH>
4217                        <IPARAMVALUE NAME="ClassName">
4218                            <CLASSNAME NAME="CIM_LogicalDisk"/>
4219                        </IPARAMVALUE>
```

```
4220              <IPARAMVALUE NAME="LocalOnly"><VALUE>TRUE</VALUE></IPARAMVALUE>
4221              <IPARAMVALUE NAME="DeepInheritance"><VALUE>TRUE</VALUE></IPARAMVALUE>
4222          </IMETHODCALL>
4223        </SIMPLEREQ>
4224      </MESSAGE>
4225    </CIM>
```

4226 Following is an HTTP response to the preceding request indicating success of the preceding operation,
4227 returning the requested instances.

```
4228    HTTP/1.1 200 OK
4229    Content-Type: application/xml; charset=utf-8
4230    Content-Length: xxxx
4231    Ext:
4232    Cache-Control: no-cache
4233    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4234    73-CIMOperation: MethodResponse
4235
4236    <?xml version="1.0" encoding="utf-8" ?>
4237    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4238      <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4239        <SIMPLERSP>
4240          <IMETHODRESPONSE NAME="EnumerateInstances">
4241            <IRETURNVALUE>
4242              <VALUE.NAMEDINSTANCE>
4243                <INSTANCENAME CLASSNAME="Erewhon_LogicalDisk">
4244                  ...
4245                </INSTANCENAME>
4246                <INSTANCE CLASSNAME="Erewhon_LogicalDisk">
4247                  ...
4248                </INSTANCE>
4249              </VALUE.NAMEDINSTANCE>
4250              ...
4251              <VALUE.NAMEDINSTANCE>
4252                <INSTANCENAME CLASSNAME="Foobar_LogicalDisk">
4253                  ...
4254                </INSTANCENAME>
4255                <INSTANCE CLASSNAME="Foobar_LogicalDisk">
4256                  ...
4257                </INSTANCE>
4258              </VALUE.NAMEINSTANCE>
4259            </IRETURNVALUE>
4260          </IMETHODRESPONSE>
4261        </SIMPLERSP>
4262      </MESSAGE>
4263    </CIM>
```

4264 A.9 Retrieval of a Single Property

4265 The following HTTP request illustrates how a client retrieves the FreeSpace property from the instance
4266 MyDisk.DeviceID="C:". This example demonstrates how to use the GetInstance operation with a property
4267 list filter instead of the deprecated GetProperty operation.

```
4268    M-POST /cimom HTTP/1.1
```

```
4269        HOST: http://www.myhost.com/
4270        Content-Type: application/xml; charset=utf-8
4271        Content-Length: xxxx
4272        Man: http://www.dmtf.org/cim/operation ; ns=73
4273        73-CIMOperation: MethodCall
4274        73-CIMMethod: GetInstance
4275        73-CIMObject: root%2FmyNamespace
4276
4277        <?xml version="1.0" encoding="utf-8" ?>
4278        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4279          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4280            <SIMPLEREQ>
4281              <IMETHODCALL NAME="GetInstance">
4282                <LOCALNAMESPACEPATH>
4283                  <NAMESPACE NAME="root"/>
4284                  <NAMESPACE NAME="myNamespace"/>
4285                </LOCALNAMESPACEPATH>
4286                <IPARAMVALUE NAME="InstanceName">
4287                  <INSTANCENAME CLASSNAME="MyDisk">
4288                    <KEYBINDING NAME="DeviceID">
4289                      <KEYVALUE>C:</KEYVALUE>
4290                    </KEYBINDING>
4291                  </INSTANCENAME>
4292                </IPARAMVALUE>
4293                <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
4294                <IPARAMVALUE NAME="PropertyList">
4295                  <VALUE>FreeSpace</VALUE>
4296                </IPARAMVALUE>
4297              </IMETHODCALL>
4298            </SIMPLEREQ>
4299          </MESSAGE>
4300        </CIM>
```

4301 Following is an HTTP response to the preceding request indicating success of the preceding operation,
4302 returning the requested instance with the requested property value.

```
4303        HTTP/1.1 200 OK
4304        Content-Type: application/xml; charset=utf-8
4305        Content-Length: xxxx
4306        Ext:
4307        Cache-Control: no-cache
4308        Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4309        73-CIMOperation: MethodResponse
4310
4311        <?xml version="1.0" encoding="utf-8" ?>
4312        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4313          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4314            <SIMPLERSP>
4315              <IMETHODRESPONSE NAME="GetInstance">
4316                <IRETURNVALUE>
4317                  <INSTANCE CLASSNAME="Erewhon_LogicalDisk">
4318                    <PROPERTY NAME="FreeSpace" TYPE="uint32">
4319                      <VALUE>6752332</VALUE>
4320                    </PROPERTY>
```

```
4321                    </INSTANCE>
4322                  </IRETURNVALUE>
4323                </IMETHODRESPONSE>
4324              </SIMPLERSP>
4325            </MESSAGE>
4326        </CIM>
```

4327 A.10 Execution of an Extrinsic Method

4328 The following HTTP request illustrates how a client executes the SetPowerState method on the instance
4329 MyDisk.DeviceID="C:".

```
4330        M-POST /cimom HTTP/1.1
4331        HOST: http://www.myhost.com/
4332        Content-Type: application/xml; charset=utf-8
4333        Content-Length: xxxx
4334        Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4335        73-CIMOperation: MethodCall
4336        73-CIMMethod: SetPowerState
4337        73-CIMObject: root%2FmyNamespace%3AMyDisk.Name%3D%22C%3A%22
4338
4339        <?xml version="1.0" encoding="utf-8" ?>
4340        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4341          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4342            <SIMPLEREQ>
4343              <METHODCALL NAME="SetPowerState">
4344                <LOCALINSTANCEPATH>
4345                  <LOCALNAMESPACEPATH>
4346                    <NAMESPACE NAME="root"/>
4347                    <NAMESPACE NAME="myNamespace"/>
4348                  </LOCALNAMESPACEPATH>
4349                  <INSTANCENAME CLASSNAME="MyDisk">
4350                    <KEYBINDING NAME="Name"><KEYVALUE>C:</KEYVALUE></KEYBINDING>
4351                  </INSTANCENAME>
4352                </LOCALINSTANCEPATH>
4353                <PARAMVALUE NAME="PowerState"><VALUE>1</VALUE></PARAMVALUE>
4354                <PARAMVALUE NAME="Time">
4355                  <VALUE>00000001132312.000000:000</VALUE>
4356                </PARAMVALUE>
4357              </METHODCALL>
4358            </SIMPLEREQ>
4359          </MESSAGE>
4360        </CIM>
```

4361 Following is an HTTP response to the preceding request indicating the success of the preceding
4362 operation.

```
4363        HTTP/1.1 200 OK
4364        Content-Type: application/xml; charset=utf-8
4365        Content-Length: xxxx
4366        Ext:
4367        Cache-Control: no-cache
4368        Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4369        73-CIMOperation: MethodResponse
```

```
4370
4371        <?xml version="1.0" encoding="utf-8" ?>
4372        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4373          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4374            <SIMPLERSP>
4375              <METHODRESPONSE NAME="SetPowerState">
4376                <RETURNVALUE>
4377                    <VALUE>0</VALUE>
4378                </RETURNVALUE>
4379              </METHODRESPONSE>
4380            </SIMPLERSP>
4381          </MESSAGE>
4382        </CIM>
```

4383 ## A.11 Indication Delivery Example

4384 The following HTTP request illustrates the format for sending an indication of type CIM_AlertIndication to
4385 a WBEM listener.

```
4386        M-POST /cimlistener/browser HTTP/1.1
4387        HOST: http://www.acme.com/
4388        Content-Type: application/xml; charset=utf-8
4389        Content-Length: XXX
4390        Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=40
4391        40-CIMExport: MethodRequest
4392        40-CIMExportMethod: ExportIndication
4393
4394        <?xml version="1.0" encoding="utf-8" ?>
4395        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4396          <MESSAGE ID="1007" PROTOCOLVERSION="1.0">
4397            <SIMPLEEXPREQ>
4398              <EXPMETHODCALL NAME="ExportIndication">
4399                <EXPPARAMVALUE NAME="NewIndication">
4400                  <INSTANCE CLASSNAME="CIM_AlertIndication" >
4401                    <PROPERTY NAME="Description" TYPE="string">
4402                      <VALUE>Sample CIM_AlertIndication indication</VALUE>
4403                    </PROPERTY>
4404                    <PROPERTY NAME="AlertType" TYPE="uint16">
4405                      <VALUE>1</VALUE>
4406                    </PROPERTY>
4407                    <PROPERTY NAME="PerceivedSeverity" TYPE="uint16">
4408                      <VALUE>3</VALUE>
4409                    </PROPERTY>
4410                    <PROPERTY NAME="ProbableCause" TYPE="uint16">
4411                      <VALUE>2</VALUE>
4412                    </PROPERTY>
4413                    <PROPERTY NAME="IndicationTime" TYPE="datetime">
4414                      <VALUE>20010515104354.000000:000</VALUE>
4415                    </PROPERTY>
4416                  </INSTANCE>
4417                </EXPPARAMVALUE>
4418              </EXPMETHODCALL>
4419            </SIMPLEEXPREQ>
```

```
4420        </MESSAGE>
4421      </CIM>
```

4422 Following is an HTTP response to the preceding request indicating a successful receipt by the WBEM
4423 listener.

```
4424      HTTP/1.1 200 OK
4425      Content-Type: application/xml; charset=utf-8
4426      Content-Length: 267
4427      Ext:
4428      Cache-Control: no-cache
4429      Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=40
4430      40-CIMExport: MethodResponse
4431
4432      <?xml version="1.0" encoding="utf-8" ?>
4433      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4434        <MESSAGE ID="1007" PROTOCOLVERSION="1.0">
4435          <SIMPLEEXPRSP>
4436            <EXPMETHODRESPONSE NAME="ExportIndication">
4437              <IRETURNVALUE></IRETURNVALUE>
4438            </EXPMETHODRESPONSE>
4439          </SIMPLEEXPRSP>
4440        </MESSAGE>
4441      </CIM>
```

4442 **A.12 Subscription Example**

4443 A WBEM client application activates a subscription by creating an instance of the
4444 CIM_IndicationSubscription class, which defines an association between a CIM_IndicationFilter (a filter)
4445 instance and a CIM_IndicationHandler (a handler) instance. The CIM_IndicationFilter instance defines the
4446 filter criteria and data project list to describe the desired indication stream. The CIM_IndicationHandler
4447 instance defines the desired indication encoding, destination location, and protocol for delivering the
4448 indication stream.

4449 The following HTTP request illustrates how a client creates an instance of the class CIM_IndicationFilter.
4450 Note that the exact syntax of the WMI Query Language is still under review and is subject to change.

```
4451      Host: bryce
4452      Content-Type: application/xml; charset=utf-8
4453      Content-Length: XXXX
4454      Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=20
4455      20-CIMProtocolVersion: 1.0
4456      20-CIMOperation: MethodCall
4457      20-CIMMethod: CreateInstance
4458      20-CIMObject: root/cimv2
4459
4460      <?xml version="1.0" encoding="utf-8"?>
4461      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4462        <MESSAGE ID="53000" PROTOCOLVERSION="1.0">
4463          <SIMPLEREQ>
4464            <IMETHODCALL NAME="CreateInstance">
4465              <LOCALNAMESPACEPATH>
4466                <NAMESPACE NAME="root"/>
4467                <NAMESPACE NAME="cimv2"/>
4468              </LOCALNAMESPACEPATH>
```

```
4469                    <IPARAMVALUE NAME="NewInstance">
4470                       <INSTANCE CLASSNAME="CIM_IndicationFilter">
4471                          <PROPERTY NAME="SystemCreationClassName" TYPE="string">
4472                             <VALUE>CIM_UnitaryComputerSystem</VALUE>
4473                          </PROPERTY>
4474                          <PROPERTY NAME="SystemName" TYPE="string">
4475                             <VALUE>server001.acme.com</VALUE>
4476                          </PROPERTY>
4477                          <PROPERTY NAME="CreationClassName" TYPE="string">
4478                             <VALUE>CIM_IndicationFilter</VALUE>
4479                          </PROPERTY>
4480                          <PROPERTY NAME="Name" TYPE="string">
4481                             <VALUE>ACMESubscription12345</VALUE>
4482                          </PROPERTY>
4483                          <PROPERTY NAME="SourceNamespace" TYPE="string">
4484                             <VALUE>root/cimv2</VALUE>
4485                          </PROPERTY>
4486                          <PROPERTY NAME="Query" TYPE="string">
4487                             <VALUE>
4488                             SELECT Description, AlertType, PerceivedSeverity,
4489                                 ProbableCause, IndicationTime
4490                                 FROM CIM_AlertIndication
4491                                 WHERE PerceivedSeverity = 3
4492                             </VALUE>
4493                          </PROPERTY>
4494                          <PROPERTY NAME="QueryLanguage" TYPE="string">
4495                             <VALUE>WQL</VALUE>
4496                          </PROPERTY>
4497                       </INSTANCE>
4498                    </IPARAMVALUE>
4499                 </IMETHODCALL>
4500              </SIMPLEREQ>
4501           </MESSAGE>
4502        </CIM>
```

4503   Following is an HTTP response to the preceding request indicating success of the preceding operation.

```
4504        HTTP/1.1 200 OK
4505        Content-Type: application/xml; charset=utf-8
4506        Content-Length: XXX
4507        Ext:
4508        Cache-Control: no-cache
4509        Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=28
4510        28-CIMOperation: MethodResponse
4511
4512        <?xml version="1.0" encoding="utf-8" ?>
4513        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4514           <MESSAGE ID="53000" PROTOCOLVERSION="1.0">
4515              <SIMPLERSP>
4516                 <IMETHODRESPONSE NAME="CreateInstance">
4517                    <IRETURNVALUE>
4518                       <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
4519                          <KEYBINDING NAME="SystemCreationClassName">
4520                             <KEYVALUE VALUETYPE="string">
```

```
4521                                   CIM_UnitaryComputerSystem
4522                                </KEYVALUE>
4523                             </KEYBINDING>
4524                             <KEYBINDING NAME="SystemName">
4525                                <KEYVALUE VALUETYPE="string">
4526                                   server001.acme.com
4527                                </KEYVALUE>
4528                             </KEYBINDING>
4529                             <KEYBINDING NAME="CreationClassName">
4530                                <KEYVALUE VALUETYPE="string">
4531                                   CIM_IndicationFilter
4532                                </KEYVALUE>
4533                             </KEYBINDING>
4534                             <KEYBINDING NAME="Name">
4535                                <KEYVALUE VALUETYPE="string">
4536                                   ACMESubscription12345
4537                                </KEYVALUE>
4538                             </KEYBINDING>
4539                          </INSTANCENAME>
4540                       </IRETURNVALUE>
4541                    </IMETHODRESPONSE>
4542                 </SIMPLERSP>
4543              </MESSAGE>
4544           </CIM>
```

4545 The following HTTP request illustrates how a client creates an instance of the class
4546 CIM_IndicationHandlerCIMXML.

```
4547        M-POST /cimom HTTP/1.1
4548        Host: bryce
4549        Content-Type: application/xml; charset=utf-8
4550        Content-Length: XXX
4551        Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=20
4552        20-CIMProtocolVersion: 1.0
4553        20-CIMOperation: MethodCall
4554        20-CIMMethod: CreateInstance
4555        20-CIMObject: root/cimv2
4556
4557        <?xml version="1.0" encoding="utf-8"?>
4558        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4559           <MESSAGE ID="54000" PROTOCOLVERSION="1.0">
4560              <SIMPLEREQ>
4561                 <IMETHODCALL NAME="CreateInstance">
4562                    <LOCALNAMESPACEPATH>
4563                       <NAMESPACE NAME="root"/>
4564                       <NAMESPACE NAME="cimv2"/>
4565                    </LOCALNAMESPACEPATH>
4566                    <IPARAMVALUE NAME="NewInstance">
4567                       <INSTANCE CLASSNAME="CIM_IndicationHandlerCIMXML">
4568                          <PROPERTY NAME="SystemCreationClassName" TYPE="string">
4569                             <VALUE>CIM_UnitaryComputerSystem</VALUE>
4570                          </PROPERTY>
4571                          <PROPERTY NAME="SystemName" TYPE="string">
4572                             <VALUE>server001.acme.com</VALUE>
```

```
4573                              </PROPERTY>
4574                              <PROPERTY NAME="CreationClassName" TYPE="string">
4575                                 <VALUE>CIM_IndicationHandlerCIMXML</VALUE>
4576                              </PROPERTY>
4577                              <PROPERTY NAME="Name" TYPE="string">
4578                                 <VALUE>ACMESubscription12345</VALUE>
4579                              </PROPERTY>
4580                              <PROPERTY NAME="Owner" TYPE="string">
4581                                 <VALUE>ACMEAlertMonitoringConsole</VALUE>
4582                              </PROPERTY>
4583                              <PROPERTY NAME="Destination" TYPE="string">
4584                                 <VALUE>HTTP://www.acme.com/cimlistener/browser</VALUE>
4585                              </PROPERTY>
4586                           </INSTANCE>
4587                        </IPARAMVALUE>
4588                     </IMETHODCALL>
4589                  </SIMPLEREQ>
4590             </MESSAGE>
4591        </CIM>
```

4592   Following is an HTTP response to the preceding request indicating the success of the preceding
4593   operation.

```
4594        HTTP/1.1 200 OK
4595        Content-Type: application/xml; charset=utf-8
4596        Content-Length: XXX
4597        Ext:
4598        Cache-Control: no-cache
4599        Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=27
4600        27-CIMOperation: MethodResponse
4601
4602        <?xml version="1.0" encoding="utf-8" ?>
4603        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4604           <MESSAGE ID="54000" PROTOCOLVERSION="1.0">
4605              <SIMPLERSP>
4606                 <IMETHODRESPONSE NAME="CreateInstance">
4607                    <IRETURNVALUE>
4608                       <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
4609                          <KEYBINDING NAME="SystemCreationClassName">
4610                             <KEYVALUE VALUETYPE="string">
4611                                CIM_UnitaryComputerSystem
4612                             </KEYVALUE>
4613                          </KEYBINDING>
4614                          <KEYBINDING NAME="SystemName">
4615                             <KEYVALUE VALUETYPE="string">
4616                                server001.acme.com
4617                             </KEYVALUE>
4618                          </KEYBINDING>
4619                          <KEYBINDING NAME="CreationClassName">
4620                             <KEYVALUE VALUETYPE="string">
4621                                CIM_IndicationHandlerCIMXML
4622                             </KEYVALUE>
4623                          </KEYBINDING>
4624                          <KEYBINDING NAME="Name">
```

```
4625                            <KEYVALUE VALUETYPE="string">
4626                                ACMESubscription12345
4627                            </KEYVALUE>
4628                        </KEYBINDING>
4629                    </INSTANCENAME>
4630                </IRETURNVALUE>
4631            </IMETHODRESPONSE>
4632        </SIMPLERSP>
4633      </MESSAGE>
4634    </CIM>
```

4635   The following HTTP request illustrates how a client creates an instance of the class
4636   CIM_IndicationSubscription.

```
4637    M-POST /cimom HTTP/1.1
4638    Host: bryce
4639    Content-Type: application/xml; charset=utf-8
4640    Content-Length: XXXX
4641    Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=55
4642    55-CIMProtocolVersion: 1.0
4643    55-CIMOperation: MethodCall
4644    55-CIMMethod: CreateInstance
4645    55-CIMObject: root/cimv2
4646
4647    <?xml version="1.0" encoding="utf-8"?>
4648    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4649      <MESSAGE ID="55000" PROTOCOLVERSION="1.0">
4650        <SIMPLEREQ>
4651          <IMETHODCALL NAME="CreateInstance">
4652            <LOCALNAMESPACEPATH>
4653              <NAMESPACE NAME="root"/>
4654              <NAMESPACE NAME="cimv2"/>
4655            </LOCALNAMESPACEPATH>
4656            <IPARAMVALUE NAME="NewInstance">
4657              <INSTANCE CLASSNAME="CIM_IndicationSubscription">
4658                <PROPERTY.REFERENCE NAME="Filter"
4659                                    REFERENCECLASS="CIM_IndicationFilter">
4660                  <VALUE.REFERENCE>
4661                    <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
4662                      <KEYBINDING NAME="SystemCreationClassName">
4663                        <KEYVALUE VALUETYPE="string">
4664                            CIM_UnitaryComputerSystem
4665                        </KEYVALUE>
4666                      </KEYBINDING>
4667                      <KEYBINDING NAME="SystemName">
4668                        <KEYVALUE VALUETYPE="string">
4669                            server001.acme.com
4670                        </KEYVALUE>
4671                      </KEYBINDING>
4672                      <KEYBINDING NAME="CreationClassName">
4673                        <KEYVALUE VALUETYPE="string">
4674                            CIM_IndicationFilter
4675                        </KEYVALUE>
4676                      </KEYBINDING>
```

```
4677                              <KEYBINDING NAME="Name">
4678                                  <KEYVALUE VALUETYPE="string">
4679                                      ACMESubscription12345
4680                                  </KEYVALUE>
4681                              </KEYBINDING>
4682                          </INSTANCENAME>
4683                      </VALUE.REFERENCE>
4684                  </PROPERTY.REFERENCE>
4685                  <PROPERTY.REFERENCE NAME="Handler"
4686                                      REFERENCECLASS="CIM_IndicationHandler">
4687                      <VALUE.REFERENCE>
4688                          <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
4689                              <KEYBINDING NAME="SystemCreationClassName">
4690                                  <KEYVALUE VALUETYPE="string">
4691                                      CIM_UnitaryComputerSystem
4692                                  </KEYVALUE>
4693                              </KEYBINDING>
4694                              <KEYBINDING NAME="SystemName">
4695                                  <KEYVALUE VALUETYPE="string">
4696                                      server001.acme.com
4697                                  </KEYVALUE>
4698                              </KEYBINDING>
4699                              <KEYBINDING NAME="CreationClassName">
4700                                  <KEYVALUE VALUETYPE="string">
4701                                      CIM_IndicationHandlerCIMXML
4702                                  </KEYVALUE>
4703                              </KEYBINDING>
4704                              <KEYBINDING NAME="Name">
4705                                  <KEYVALUE VALUETYPE="string">
4706                                      ACMESubscription12345
4707                                  </KEYVALUE>
4708                              </KEYBINDING>
4709                          </INSTANCENAME>
4710                      </VALUE.REFERENCE>
4711                  </PROPERTY.REFERENCE>
4712              </INSTANCE>
4713          </IPARAMVALUE>
4714      </IMETHODCALL>
4715    </SIMPLEREQ>
4716  </MESSAGE>
4717 </CIM>
```

4718 Following is an HTTP response to the preceding request indicating the success of the preceding
4719 operation.

```
4720  HTTP/1.1 200 OK
4721  Content-Type: application/xml; charset=utf-8
4722  Content-Length: XXXX
4723  Ext:
4724  Cache-Control: no-cache
4725  Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=75
4726  75-CIMOperation: MethodResponse
4727
4728  <?xml version="1.0" encoding="utf-8" ?>
```

```
4729        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4730          <MESSAGE ID="55000" PROTOCOLVERSION="1.0">
4731            <SIMPLERSP>
4732              <IMETHODRESPONSE NAME="CreateInstance">
4733                <IRETURNVALUE>
4734                  <INSTANCENAME CLASSNAME="CIM_IndicationSubscription">
4735                    <KEYBINDING NAME="Filter">
4736                      <VALUE.REFERENCE>
4737                        <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
4738                          <KEYBINDING NAME="SystemCreationClassName">
4739                            <KEYVALUE VALUETYPE="string">
4740                              CIM_UnitaryComputerSystem
4741                            </KEYVALUE>
4742                          </KEYBINDING>
4743                          <KEYBINDING NAME="SystemName">
4744                            <KEYVALUE VALUETYPE="string">
4745                              server001.acme.com
4746                            </KEYVALUE>
4747                          </KEYBINDING>
4748                          <KEYBINDING NAME="CreationClassName">
4749                            <KEYVALUE VALUETYPE="string">
4750                              CIM_IndicationFilter
4751                            </KEYVALUE>
4752                          </KEYBINDING>
4753                          <KEYBINDING NAME="Name">
4754                            <KEYVALUE VALUETYPE="string">
4755                              ACMESubscription12345
4756                            </KEYVALUE>
4757                          </KEYBINDING>
4758                        </INSTANCENAME>
4759                      </VALUE.REFERENCE>
4760                    </KEYBINDING>
4761                    <KEYBINDING NAME="Handler">
4762                      <VALUE.REFERENCE>
4763                        <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
4764                          <KEYBINDING NAME="SystemCreationClassName">
4765                            <KEYVALUE VALUETYPE="string">
4766                              CIM_UnitaryComputerSystem
4767                            </KEYVALUE>
4768                          </KEYBINDING>
4769                          <KEYBINDING NAME="SystemName">
4770                            <KEYVALUE VALUETYPE="string">
4771                             server001.acme.com
4772                            </KEYVALUE>
4773                          </KEYBINDING>
4774                          <KEYBINDING NAME="CreationClassName">
4775                            <KEYVALUE VALUETYPE="string">
4776                              CIM_IndicationHandlerCIMXML
4777                            </KEYVALUE>
4778                          </KEYBINDING>
4779                          <KEYBINDING NAME="Name">
4780                            <KEYVALUE VALUETYPE="string">
4781                              ACMESubscription12345
```

```
4782                                    </KEYVALUE>
4783                                 </KEYBINDING>
4784                              </INSTANCENAME>
4785                           </VALUE.REFERENCE>
4786                        </KEYBINDING>
4787                     </INSTANCENAME>
4788                  </IRETURNVALUE>
4789               </IMETHODRESPONSE>
4790            </SIMPLERSP>
4791         </MESSAGE>
4792      </CIM>
```

4793 A.13 Multiple Operations Example

4794 The following HTTP request illustrates how a client performs multiple operations. This example batches a
4795 GetClass, an EnumerateInstanceNames, and an EnumerateInstance operation on
4796 CIM_ObjectManagerAdapter.

```
4797      POST /CIMOM1 HTTP/1.1
4798      Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
4799      Content-Length: XXX
4800      Host: localhost:5988
4801      CIMOperation: MethodCall
4802      CIMProtocolVersion: 1.0
4803      Content-Type: application/xml; charset=utf-8
4804      CIMBatch: CIMBatch
4805      <?xml version="1.0" encoding="UTF-8"?>
4806
4807      <CIM DTDVERSION="2.0" CIMVERSION="2.0">
4808         <MESSAGE ID="2004:2:5:1:1:11:41:1" PROTOCOLVERSION="1.0">
4809            <MULTIREQ>
4810               <SIMPLEREQ>
4811                  <IMETHODCALL NAME="GetClass">
4812                     <LOCALNAMESPACEPATH>
4813                        <NAMESPACE NAME="interop" />
4814                     </LOCALNAMESPACEPATH>
4815                     <IPARAMVALUE NAME="ClassName">
4816                        <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
4817                     </IPARAMVALUE>
4818                     <IPARAMVALUE NAME="LocalOnly">
4819                        <VALUE>FALSE</VALUE>
4820                     </IPARAMVALUE>
4821                     <IPARAMVALUE NAME="IncludeClassOrigin">
4822                        <VALUE>TRUE</VALUE>
4823                     </IPARAMVALUE>
4824                  </IMETHODCALL>
4825               </SIMPLEREQ>
4826               <SIMPLEREQ>
4827                  <IMETHODCALL NAME="Associators">
4828                     <LOCALNAMESPACEPATH>
4829                        <NAMESPACE NAME="interop" />
4830                     </LOCALNAMESPACEPATH>
4831                     <IPARAMVALUE NAME="ObjectName">
4832                        <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
```

```
4833                        </IPARAMVALUE>
4834                        <IPARAMVALUE NAME="IncludeQualifiers">
4835                            <VALUE>TRUE</VALUE>
4836                        </IPARAMVALUE>
4837                        <IPARAMVALUE NAME="IncludeClassOrigin">
4838                            <VALUE>TRUE</VALUE>
4839                        </IPARAMVALUE>
4840                    </IMETHODCALL>
4841                </SIMPLEREQ>
4842                <SIMPLEREQ>
4843                    <IMETHODCALL NAME="EnumerateInstanceNames">
4844                        <LOCALNAMESPACEPATH>
4845                            <NAMESPACE NAME="interop" />
4846                        </LOCALNAMESPACEPATH>
4847                        <IPARAMVALUE NAME="ClassName">
4848                            <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
4849                        </IPARAMVALUE>
4850                    </IMETHODCALL>
4851                </SIMPLEREQ>
4852                <SIMPLEREQ>
4853                    <IMETHODCALL NAME="EnumerateInstances">
4854                        <LOCALNAMESPACEPATH>
4855                            <NAMESPACE NAME="interop" />
4856                        </LOCALNAMESPACEPATH>
4857                        <IPARAMVALUE NAME="ClassName">
4858                            <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
4859                        </IPARAMVALUE>
4860                        <IPARAMVALUE NAME="LocalOnly">
4861                            <VALUE>FALSE</VALUE>
4862                        </IPARAMVALUE>
4863                    </IMETHODCALL>
4864                </SIMPLEREQ>
4865            </MULTIREQ>
4866        </MESSAGE>
4867    </CIM>
```

4868 Following is the HTTP response to the preceding request indicating the success of the preceding
4869 operation.

```
4870    HTTP/1.1 200 OK
4871    CIMOperation: MethodResponse
4872    Content-Length: XXX
4873
4874    <?xml version="1.0" encoding="UTF-8"?>
4875    <CIM DTDVERSION="2.0" CIMVERSION="2.0">
4876        <MESSAGE ID="2004:2:5:1:1:11:41:1" PROTOCOLVERSION="1.0">
4877            <MULTIRSP>
4878                <SIMPLERSP>
4879                    <IMETHODRESPONSE NAME="GetClass">
4880                        <IRETURNVALUE>
4881                            <CLASS SUPERCLASS="CIM_WBEMService"
4882                                    NAME="CIM_ObjectManagerAdapter">
4883                                ...
4884                            </CLASS>
```

```
4885                        </IRETURNVALUE>
4886                    </IMETHODRESPONSE>
4887                </SIMPLERSP>
4888                <SIMPLERSP>
4889                    <IMETHODRESPONSE NAME="Associators">
4890                        <IRETURNVALUE>
4891                            <VALUE.OBJECTWITHPATH>
4892                                ...
4893                            </VALUE.OBJECTWITHPATH>
4894                            <VALUE.OBJECTWITHPATH>
4895                                ...
4896                            </VALUE.OBJECTWITHPATH>
4897                                ...
4898                        </IRETURNVALUE>
4899                    </IMETHODRESPONSE>
4900                </SIMPLERSP>
4901                <SIMPLERSP>
4902                    <IMETHODRESPONSE NAME="EnumerateInstanceNames">
4903                        <IRETURNVALUE>
4904                            <INSTANCENAME CLASSNAME="WBEMSolutions_ObjectManagerAdapter">
4905                                ...
4906                            </INSTANCENAME>
4907                            <INSTANCENAME CLASSNAME="WBEMSolutions_ObjectManagerAdapter">
4908                                ...
4909                            </INSTANCENAME>
4910                                ...
4911                        </IRETURNVALUE>
4912                    </IMETHODRESPONSE>
4913                </SIMPLERSP>
4914                <SIMPLERSP>
4915                    <IMETHODRESPONSE NAME="EnumerateInstances">
4916                        <IRETURNVALUE>
4917                            <VALUE.NAMEDINSTANCE>
4918                                ...
4919                            </VALUE.NAMEDINSTANCE>
4920                            <VALUE.NAMEDINSTANCE>
4921                                ...
4922                            </VALUE.NAMEDINSTANCE>
4923                                ...
4924                        </IRETURNVALUE>
4925                    </IMETHODRESPONSE>
4926                </SIMPLERSP>
4927            </MULTIRSP>
4928        </MESSAGE>
4929    </CIM>
```

<div style="text-align: center;">

4930 **ANNEX B**

4931 (informative)

4932

4933

4934 **LocalOnly Parameter Discussion**

</div>

4935 This annex discusses the issues associated with the 1.1 definition of the `LocalOnly` parameter for the
4936 GetInstance and EnumerateInstances operations.

B.1   Explanation of the Deprecated 1.1 Interpretation

4938 In April 2002, two DMTF Change Requests (CRs), CR809 (EnumerateInstances) and CR815
4939 (GetInstance), were approved and incorporated into version 1.1of this document to clarify the
4940 interpretation of the `LocalOnly` flag for the GetInstance and EnumerateInstances operations. With these
4941 CRs, the definition of the `LocalOnly` flag for these operations was modified to align with the
4942 interpretation of this flag for the GetClass and EnumerateClasses operations. This change was incorrect,
4943 resulted in reduced functionality, and introduced several backward compatibility issues.

4944 To clarify the difference between the 1.0 Interpretation and the 1.1 Interpretation (CR815), consider the
4945 following example:

```
4946    class A {
4947          [Key]
4948       string name;
4949       uint32 counter = 3;
4950    };
4951
4952    class B : A {
4953       uint32 moreData = 4;
4954    };
4955
4956    instance of A {
4957       name = "Roger";
4958    };
4959
4960    instance of B {
4961       name = "Karl";
4962       counter = 3;
4963       moreData = 5;
4964    };
4965
4966    instance of B {
4967       name = "Denise";
4968       counter = 5;
4969    };
```

4970   Assuming `PropertyList = NULL` and `LocalOnly = TRUE`, Table B-1 shows the properties returned
4971   by a GetInstance operation.

4972   **Table B-1 – 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

4973   The properties returned using the 1.0 interpretation are consistent with the properties specified in the
4974   MOF instance definitions, and the properties returned using the 1.1 Interpretation are consistent with the
4975   properties defined in the class definitions.

B.2   Risks of Using the 1.1 Interpretation

4977   The risks of using the 1.1 interpretation are as follows:

4978   1)   Within the DMTF, promoting a property from a class to one of its superclasses is defined as a
4979   backward-compatible change that can be made in a minor revision of the CIM schema. With the 1.1
4980   interpretation, promoting a property to a superclass can cause backward-incompatible changes.

4981   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;
};
```

4991   Now suppose that the schema is modified in version 1.1 to promote the property moreData from
4992   class B to class A.

```
class A {
      [Key]
   string name;
   uint32 counter = 3;
   uint32 moreData = 4;
};

class B : A {
};
```

5002   Using these examples, Table B-2 shows the properties returned by a call to GetInstance with
5003   `PropertyList = NULL` and `LocalOnly = TRUE`. With the 1.1 Interpretation, this schema
5004   change would affect the list of properties returned. When dealing with a WBEM server that complies
5005   with the 1.1 interpretation, applications must be designed to treat "promoting properties" as a
5006   backward-compatible change.

5007 **Table B-2 – 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

5008

5009 2) The 1.1 Interpretation encourages application developers to use multiple operations to retrieve the
5010 properties of an instance. That is, a commonly-stated use model for the 1.1 interpretation is to
5011 selectively traverse subclasses getting additional properties of an instance. This practice significantly
5012 increases the risk that a client will construct an inconsistent instance. With both Interpretations,
5013 applications should be designed to ensure that dependent properties are retrieved together.

5014 ## B.3 Techniques for Differentiating between the 1.0 Interpretation and 1.1
5015 Interpretation

5016 For concrete classes, WBEM servers that comply with the 1.0 Interpretation return the value of all KEY
5017 properties not explicitly excluded by the `PropertyList` parameter. WBEM servers that comply with the
5018 1.1 interpretation return only the value of KEY properties explicitly defined in the class. Applications can
5019 use this difference to detect which interpretation is supported by a WBEM server.

5020
5021

ANNEX C
(normative)

5022
5023

Generic Operations Mapping

5025 This annex defines a mapping of generic operations (see DSP0223) to the CIM-XML protocol described
5026 in this document.

5027 A main purpose of this mapping is to support the implementations of DMTF management profiles that
5028 define operations in terms of generic operations, by providing them a translation from the generic
5029 operation listed in the management profile, to the CIM-XML operation that actually needs to be
5030 implemented.

C.1   Operations

5032 This subclause defines for each generic operation, which CIM-XML operation needs to be supported in
5033 order to support the respective generic operation.

5034 Table C-1 lists the generic operations defined in DSP0223 and for each of them, lists the name of the
5035 corresponding CIM-XML operation and a link to the description subclause.

5036   **Table C-1 – 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
EnumerateInstances	EnumerateInstances	See C.1.5
EnumerateInstanceNames	EnumerateInstanceNames	See C.1.6
Associators	Associators (ObjectName is an instance path)	See C.1.7
AssociatorNames	AssociatorNames (ObjectName is an instance path)	See C.1.8
References	References (ObjectName is an instance path)	See C.1.9
ReferenceNames	ReferenceNames (ObjectName is an instance path)	See C.1.10
OpenEnumerateInstances	OpenEnumerateInstances	See C.1.11
OpenEnumerateInstancePaths	OpenEnumerateInstancePaths	See C.1.12
OpenAssociators	OpenAssociatorInstances	See C.1.13
OpenAssociatorPaths	OpenAssociatorInstanceNames	See C.1.14
OpenReferences	OpenReferenceInstances	See C.1.15
OpenReferencePaths	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
EnumerateClasses	EnumerateClasses (ClassName is NULL)	See C.1.29
EnumerateClassNames	EnumerateClassNames (ClassName is NULL)	See C.1.30
GetSubClassesWithPath	EnumerateClasses (ClassName is non-NULL)	See 1.1.1.1.1.1A.1.1
GetSubClassPaths	EnumerateClassNames (ClassName is non-NULL)	See 1.1.1.1.1.1A.1.1
AssociatorClasses	Associators (ObjectName is a class path)	See C.1.31
AssociatorClassPaths	AssociatorNames (ObjectName is a class path)	See C.1.32
ReferenceClasses	References (ObjectName is a class path)	See C.1.33
ReferenceClassPaths	ReferenceNames (ObjectName is a class path)	See C.1.34
GetQualifierType	GetQualifier	See C.1.35
DeleteQualifierType	DeleteQualifier	See C.1.36
ModifyQualifierType	SetQualifier (Qualifier exists)	See C.1.37
CreateQualifierType	SetQualifier (Qualifier does not exist)	See C.1.38
EnumerateQualifierTypesWithPath	EnumerateQualifiers	See C.1.39

5037   In the following subclauses, the CIM-XML Type listed in the tables is either an intrinsic CIM type (e.g.
5038   "boolan"), or one of the pseudo-types defined in this document (e.g. "instanceName").

5039   **C.1.1   GetInstance**

5040   **CIM-XML Operation Name:** GetInstance

5041   **Purpose:** Retrieve an instance given its instance path.

5042   **Operation Input Parameters:**

5043

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	
N/A	N/A	IncludeQualifiers	boolean	See 2)
N/A	N/A	LocalOnly	boolean	See 3)

5044   1)   The CIM-XML parameter *InstanceName* includes the model path portion of the instance path of the
5045   instance. The generic parameter *InstancePath* corresponds to the combination of the CIM-XML
5046   parameter *InstanceName* and the target namespace of the CIM-XML operation.

5047   2)   The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The
5048   defined behavior of generic operation *GetInstance* conforms to the behavior of CIM-XML operation
5049   *GetInstance* with *IncludeQualifiers=false*, which is the recommended value to be used for CIM-XML
5050   clients since version 1.2 of this document.

5051   3)   The CIM-XML parameter *LocalOnly* has been deprecated in version 1.2 of this document. The
5052   defined behavior of generic operation *GetInstance* conforms to the behavior of CIM-XML operation
5053   *GetInstance* with *LocalOnly=false*, which is the recommended value to be used for CIM-XML clients
5054   since version 1.2 of this document.

5055   **Operation Output Parameters:**

5056

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
Instance	InstanceSpecification	return value	instance	

5057   **Optional behavior:**

5058   • CIM-XML allows implementations to optimize by not including properties in the returned
5059   instance that have a value of NULL.

5060   **Deviations:** None

5061   **C.1.2   DeleteInstance**

5062   **CIM-XML Operation Name:** DeleteInstance

5063   **Purpose:** Delete an instance given its instance path.

5064   **Operation Input Parameters:**

5065

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	target namespace	N/A	See 1)
		InstanceName	instanceName	See 1)

5066   1)   The CIM-XML parameter *InstanceName* includes the model path portion of the instance path of the
5067   instance. The generic parameter *InstancePath* corresponds to the combination of the CIM-XML
5068   parameter *InstanceName* and the target namespace of the CIM-XML operation.

5069   **Operation Output Parameters:** None

5070   **Deviations:** None

5071   C.1.3   ModifyInstance

5072   **CIM-XML Operation Name:** ModifyInstance

5073   **Purpose:** Modify property values of an instance given its instance path.

5074   **Operation Input Parameters:**

5075

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)

5076   1)   The CIM-XML parameter *ModifiedInstance* includes the model path portion of the instance path of
5077   the instance that is being modified, and the modified property values. The combination of the model
5078   path portion of the CIM-XML parameter *ModifiedInstance* and the target namespace of the CIM-XML
5079   operation corresponds to the generic parameter *InstancePath*.

5080   2)   The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The
5081   defined behavior of generic operation *ModifyInstance* conforms to the behavior of CIM-XML
5082   operation *ModifyInstance* with *IncludeQualifiers=false*, which is the recommended behavior for CIM-
5083   XML servers since version 1.2 of this document.

5084   **Operation Output Parameters:** None

5085   **Optional behavior:**

5086   •   [DSP0223](#) permits conformant WBEM protocols to require that all properties exposed by the
5087   creation class of the instance referenced by *InstancePath* are supplied by the WBEM client with
5088   their modified values. CIM-XML does not require that, i.e. CIM-XML permits clients to supply
5089   modified values only for a subset of these properties and those not supplied are meant to be left
5090   unchanged by the operation.

5091   **Deviations:** None

5092   C.1.4   CreateInstance

5093   **CIM-XML Operation Name:** CreateInstance

5094   **Purpose:** Create a CIM instance given the class path of its creation class.

5095   **Operation Input Parameters:**

5096

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description

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	

5097 1) The generic parameter *ClassPath* corresponds to the combination of the class name specified in the
5098   CIM-XML parameter *NewInstance* and the target namespace of the CIM-XML operation.

5099 **Operation Output Parameters:**

5100

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePath	InstancePath	return value	instanceName	

5101 **Optional behavior:** None

5102 **Deviations:** None

5103 **C.1.5   EnumerateInstances**

5104 **CIM-XML Operation Name:** EnumerateInstances

5105 **Purpose:** Retrieve the instances of a given class (including instances of its subclasses). The retrieved
5106 instances include their instance paths.

5107 **Operation Input Parameters:**

5108

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)

5109 1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter
5110   *ClassName* and the target namespace of the CIM-XML operation.

5111 2) The generic parameter *ExcludeSubclassProperties* corresponds to the negated CIM-XML parameter
5112   *DeepInheritance*.

5113 3) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The
5114   defined behavior of generic operation *EnumerateInstances* conforms to the behavior of CIM-XML
5115   operation *EnumerateInstances* with *IncludeQualifiers=false*, which is the recommended value to be
5116   used for CIM-XML clients since version 1.2 of this document.

5117 4) The CIM-XML parameter *LocalOnly* has been deprecated in version 1.2 of this document. The
5118 defined behavior of generic operation *EnumerateInstances* conforms to the behavior of CIM-XML
5119 operation *EnumerateInstances* with *LocalOnly=false*, which is the recommended value to be used for
5120 CIM-XML clients since version 1.2 of this document.

5121 5) The CIM-XML parameter IncludeClassOrigin has been deprecated in version 1.4 of this document.
5122 The defined behavior of generic operation *EnumerateInstances* conforms to the behavior of the CIM-
5123 XML operations *EnumerateInstances* with IncludeClassOrigin=false, which is the recommended
5124 value to be used for CIM-XML clients since version 1.4 of this document.

5125 6)

5126 **Operation Output Parameters:**

5127

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	namedInstance [ ]	See 1)

5128 1) The CIM-XML return value includes the set of property values including the model paths, but without
5129 namespace paths. The generic parameter *InstanceList* needs to contain the instance paths in
5130 addition to the set of property values. A CIM client side mapping layer can construct the instance
5131 paths from the model paths and the CIM-XML target namespace.

5132 **Optional behavior:**

5133 • CIM-XML allows implementations to optimize by not including properties in the returned
5134 instances that have a value of NULL.

5135 **Deviations:** None

5136 **C.1.6 EnumerateInstanceNames**

5137 **CIM-XML Operation Name:** EnumerateInstanceNames

5138 **Purpose:** Retrieve the instance paths of the instances of a given class (including instances of its
5139 subclasses).

5140 **Operation Input Parameters:**

5141

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumClassPath	ClassPath	target namespace	N/A	See 1)
		ClassName	className	See 1)

5142 1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter
5143 *ClassName* and the target namespace of the CIM-XML operation.

5144 **Operation Output Parameters:**

5145

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instanceName [ ]	See 1)

5146 1) The CIM-XML return value includes the set of model paths, but without namespace paths. The
5147   generic parameter *InstancePathList* needs to contain the instance paths, including namespace
5148   paths. A CIM client side mapping layer can construct the instance paths from the model paths and
5149   the CIM-XML target namespace.

5150 **Optional behavior:** None

5151 **Deviations:** None

5152 C.1.7 Associators

5153 **CIM-XML Operation Name:** Associators with ObjectName being an instance path

5154 **Purpose:** Retrieve the instances that are associated with a given source instance. The retrieved
5155 instances include their instance paths.

5156 **Operation Input Parameters:**

5157

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)

5158 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML
5159   parameter *ObjectName* and the target namespace of the CIM-XML operation.

5160   The generic operation *Associators* corresponds to the CIM-XML operation *Associators* when an
5161   instance path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *Associators*
5162   with a class path for its *ObjectName* parameter is covered by the generic operation
5163   *AssociatorClasses* (see C.1.31).

5164 2) The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-
5165   XML parameter. Since the defined behavior of the CIM-XML operation will result in including
5166   subclass properties, a mapping layer on the CIM client side can implement the behavior defined by
5167   the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that
5168   parameter has a value of true.

5169 3) The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The
5170   defined behavior of generic operation *Associators* conforms to the behavior of CIM-XML operation
5171   *Associators* with *IncludeQualifiers=false*, which is the recommended value to be used for CIM-XML
5172   clients since version 1.2 of this document.

5173 4) The CIM-XML parameter IncludeClassOrigin has been deprecated in version 1.4 of this document.
5174   The defined behavior of generic operation Associators conforms to the behavior of the CIM-XML
5175   operations Associators with IncludeClassOrigin=false, which is the recommended value to be used
5176   for CIM-XML clients since version 1.4 of this document.

5177 **Operation Output Parameters:**

5178

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	objectWithPath [ ]	

5179 **Optional behavior:**

5180 • CIM-XML allows implementations to optimize by not including properties in the returned
5181   instances that have a value of NULL.

5182 **Deviations:** None

5183 **C.1.8   AssociatorNames**

5184 **CIM-XML Operation Name:** AssociatorNames with ObjectName being an instance path

5185 **Purpose:** Retrieve the instance paths of the instances that are associated with a given source instance.

5186 **Operation Input Parameters:**

5187

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	

5188 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML
5189   parameter *ObjectName* and the target namespace of the CIM-XML operation.

5190   The generic operation *AssociatorNames* corresponds to the CIM-XML operation *AssociatorNames*
5191   when an instance path is passed in for its *ObjectName* parameter. Using the CIM-XML operation
5192   *AssociatorNames* with a class path for its *ObjectName* parameter is covered by the generic
5193   operation *AssociatorClassPaths* (see C.1.32).

5194 **Operation Output Parameters:**

5195

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	objectPath [ ]	

5196   **Optional behavior:** None

5197   **Deviations:** None

5198   ### C.1.9   References

5199   **CIM-XML Operation Name:** References with ObjectName being an instance path

5200   **Purpose:** Retrieve the association instances that reference a given source instance. The retrieved
5201   instances include their instance paths.

5202   **Operation Input Parameters:**

5203

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 2)
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)
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)

5204   1)   The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML
5205   parameter *ObjectName* and the target namespace of the CIM-XML operation.

5206   The generic operation *References* corresponds to the CIM-XML operation *References* when an
5207   instance path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *References*
5208   with a class path for its *ObjectName* parameter is covered by the generic operation
5209   *ReferenceClasses* (see C.1.33).

5210   2)   The CIM-XML operation *References* does not support a means to filter by class name or role name
5211   of the associated classes on the other ends of the associations referencing the source instance. The
5212   generic operation *References* does support such filtering through its parameters
5213   *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the CIM-XML
5214   operation will result in including association instances that these two parameters could filter out, a
5215   mapping layer on the CIM client side can implement the behavior defined by these two generic
5216   parameters by eliminating association instances if these filter parameters are used.

5217   3)   The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-
5218   XML parameter. Since the defined behavior of the CIM-XML operation will result in including
5219   subclass properties, a mapping layer on the CIM client side can implement the behavior defined by
5220   the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that
5221   parameter has a value of true.

5222   4)   The CIM-XML parameter *IncludeQualifiers* has been deprecated in version 1.2 of this document. The
5223   defined behavior of generic operation *References* conforms to the behavior of CIM-XML operation

5224 *References* with *IncludeQualifiers=false*, which is the recommended value to be used for CIM-XML
5225 clients since in version 1.2 of this document.

5226 5) The CIM-XML parameter IncludeClassOrigin has been deprecated in version 1.4 of this document.
5227 The defined behavior of generic operation *References* conforms to the behavior of the CIM-XML
5228 operations *References* with IncludeClassOrigin=false, which is the recommended value to be used
5229 for CIM-XML clients since version 1.4 of this document.

5230 6)

5231 **Operation Output Parameters:**

5232

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	objectWithPath [ ]	

5233 **Optional behavior:**

5234 • CIM-XML allows implementations to optimize by not including properties in the returned
5235 instances that have a value of NULL.

5236 **Deviations:** None

5237 **C.1.10 ReferenceNames**

5238 **CIM-XML Operation Name:** ReferenceNames with ObjectName being an instance path

5239 **Purpose:** Retrieve the instance paths of the association instances that reference a given source
5240 instance.

5241 **Operation Input Parameters:**

5242

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 2)
SourceRoleName	PropertyName	Role	string	
AssociatedRoleName	PropertyName	N/A	N/A	See 2)

5243 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML
5244 parameter *ObjectName* and the target namespace of the CIM-XML operation.

5245 The generic operation *ReferenceNames* corresponds to the CIM-XML operation *ReferenceNames*
5246 when an instance path is passed in for its *ObjectName* parameter. Using the CIM-XML operation
5247 *ReferenceNames* with a class path for its *ObjectName* parameter is covered by the generic
5248 operation *ReferenceClassPaths* (see C.1.34).

5249 2) The CIM-XML operation *References* does not support a means to filter by class name or role name
5250 of the associated classes on the other ends of the associations referencing the source instance. The

5251   generic operation *References* does support such filtering through its parameters
5252   *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the CIM-XML
5253   operation will result in including association instances that these two parameters could filter out, a
5254   mapping layer on the CIM client side can implement the behavior defined by these two generic
5255   parameters by eliminating association instances if these filter parameters are used.

5256   **Operation Output Parameters:**

5257

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	objectPath [ ]	

5258   **Optional behavior:** None

5259   **Deviations:** None

5260   **C.1.11 OpenEnumerateInstances**

5261   **CIM-XML Operation Name:** OpenEnumerateInstances

5262   **Purpose:** Open an enumeration session for retrieving the instances of a class (including instances of its
5263   subclasses), and optionally retrieve a first set of those instances. The retrieved instances include their
5264   instance paths.

5265   **Operation Input Parameters:**

5266

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 2)
OperationTimeout	uint32	OperationTimeout	uint32	
ContinueOnError	boolean	ContinueOnError	boolean	
MaxObjectCount	uint32	MaxObjectCount	uint32	

5267   1)   The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter
5268   *ClassName* and the target namespace of the CIM-XML operation.

5269   2)   The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-
5270   XML parameter. Since the defined behavior of the CIM-XML operation will result in including
5271   subclass properties, a mapping layer on the CIM client side can implement the behavior defined by
5272   the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that
5273   parameter has a value of true.

5274 **Operation Output Parameters:**

5275

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	

5276 **Optional behavior:**

5277 • CIM-XML allows implementations to optimize by not including properties in the returned
5278 instances that have a value of NULL.

5279 **Deviations:** None

5280 **C.1.12 OpenEnumerateInstancePaths**

5281 **CIM-XML Operation Name:** OpenEnumerateInstancePaths

5282 **Purpose:** Open an enumeration session for retrieving the instance paths of the instances of a class
5283 (including instances of its subclasses), and optionally retrieve a first set of those instance paths.

5284 **Operation Input Parameters:**

5285

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	

5286 1) The generic parameter *EnumClassPath* corresponds to the combination of the CIM-XML parameter
5287 *ClassName* and the target namespace of the CIM-XML operation.

5288 **Operation Output Parameters:**

5289

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5290 **Optional behavior:** None

5291   **Deviations:** None

5292   **C.1.13 OpenAssociators**

5293   **CIM-XML Operation Name:** OpenAssociatorInstances

5294   **Purpose:** Open an enumeration session for retrieving the instances that are associated with a given
5295   source instance, and optionally retrieve a first set of those instances. The retrieved instances include their
5296   instance paths.

5297   **Operation Input Parameters:**

5298

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	

5299   1)   The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML
5300   parameter *InstanceName* and the target namespace of the CIM-XML operation.

5301   2)   The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-
5302   XML parameter. Since the defined behavior of the CIM-XML operation will result in including
5303   subclass properties, a mapping layer on the CIM client side can implement the behavior defined by
5304   the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that
5305   parameter has a value of true.

5306   **Operation Output Parameters:**

5307

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	

5308 **Optional behavior:**

5309 • CIM-XML allows implementations to optimize by not including properties in the returned
5310 instances that have a value of NULL.

5311 **Deviations:** None

5312 **C.1.14 OpenAssociatorPaths**

5313 **CIM-XML Operation Name:** OpenAssociatorInstancePaths

5314 **Purpose:** Open an enumeration session for retrieving the instance paths of instances that are associated
5315 with a given source instance, and optionally retrieve a first set of those instance paths.

5316 **Operation Input Parameters:**

5317

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	

5318 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML
5319 parameter *InstanceName* and the target namespace of the CIM-XML operation.

5320 **Operation Output Parameters:**

5321

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5322 **Optional behavior:** None

5323 **Deviations:** None

5324   **C.1.15 OpenReferences**

5325   **CIM-XML Operation Name:** OpenReferenceInstances

5326   **Purpose:** Open an enumeration session for retrieving the association instances that reference a given
5327   source instance, and optionally retrieve a first set of those instances. The retrieved instances include their
5328   instance paths.

5329   **Operation Input Parameters:**

5330

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	

5331   1)   The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML
5332   parameter *InstanceName* and the target namespace of the CIM-XML operation.

5333   2)   The CIM-XML operation *OpenReferenceInstances* does not support a means to filter by class name
5334   or role name of the associated classes on the other ends of the associations referencing the source
5335   instance. The generic operation *OpenReferences* does support such filtering through its parameters
5336   *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the CIM-XML
5337   operation will result in including association instances that these two parameters could filter out, a
5338   mapping layer on the CIM client side can implement the behavior defined by these two generic
5339   parameters by eliminating association instances if these filter parameters are used.

5340   3)   The optional generic parameter *ExcludeSubclassProperties* does not have a corresponding CIM-
5341   XML parameter. Since the defined behavior of the CIM-XML operation will result in including
5342   subclass properties, a mapping layer on the CIM client side can implement the behavior defined by
5343   the generic parameter *ExcludeSubclassProperties* by eliminating subclass properties if that
5344   parameter has a value of true.

5345 **Operation Output Parameters:**

5346

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	

5347 **Optional behavior:**

5348 • CIM-XML allows implementations to optimize by not including properties in the returned
5349   instances that have a value of NULL.

5350 **Deviations:** None

5351 **C.1.16 OpenReferencePaths**

5352 **CIM-XML Operation Name:** OpenReferenceInstancePaths

5353 **Purpose:** Open an enumeration session for retrieving the instance paths of association instances that
5354 reference a given source instance, and optionally retrieve a first set of those instance paths.

5355 **Operation Input Parameters:**

5356

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	

5357 1) The generic parameter *SourceInstancePath* corresponds to the combination of the CIM-XML
5358   parameter *InstanceName* and the target namespace of the CIM-XML operation.

5359 2) The CIM-XML operation *OpenReferenceInstancePaths* does not support a means to filter by class
5360   name or role name of the associated classes on the other ends of the associations referencing the
5361   source instance. The generic operation *OpenReferencePaths* does support such filtering through its
5362   parameters *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the
5363   CIM-XML operation will result in including association instances that these two parameters could

5364   filter out, a mapping layer on the CIM client side can implement the behavior defined by these two
5365   generic parameters by eliminating association instances if these filter parameters are used.

5366   **Operation Output Parameters:**

5367

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5368   **Optional behavior:** None

5369   **Deviations:** None

5370   **C.1.17 OpenQueryInstances**

5371   **CIM-XML Operation Name:** OpenQueryInstances

5372   **Purpose:** Open an enumeration session for retrieving the instances representing a query result, and
5373   optionally retrieve a first set of those instances. The retrieved instances are not addressable and thus do
5374   not include any instance paths.

5375   **Operation Input Parameters:**

5376

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	

5377   **Operation Output Parameters:**

5378

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	

5379 **Optional behavior:**

5380 • CIM-XML allows implementations to optimize by not including properties in the returned
5381 instances that have a value of NULL.

5382 **Deviations:** None

5383 C.1.18 PullInstancesWithPath

5384 **CIM-XML Operation Name:** PullInstancesWithPath

5385 **Purpose:** Retrieve the next set of instances from an open enumeration session. The retrieved instances
5386 include their instance paths.

5387 **Operation Input Parameters:**

5388

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	

5389 **Operation Output Parameters:**

5390

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	

5391 **Optional behavior:**

5392 • CIM-XML allows implementations to optimize by not including properties in the returned
5393 instances that have a value of NULL.

5394 **Deviations:** None

5395 C.1.19 PullInstancePaths

5396 **CIM-XML Operation Name:** PullInstancePaths

5397 **Purpose:** Retrieve the next set of instance paths from an open enumeration session.

5398 **Operation Input Parameters:**

5399

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
MaxObjectCount	uint32	MaxObjectCount	uint32	

5400 **Operation Output Parameters:**

5401

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstancePathList	InstancePath [ ]	return value	instancePath [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5402 **Optional behavior:** None

5403 **Deviations:** None

5404 **C.1.20 PullInstances**

5405 **CIM-XML Operation Name:** PullInstances

5406 **Purpose:** Retrieve the next set of instances from an open enumeration session. The retrieved instances
5407 do not include any instance paths.

5408 **Operation Input Parameters:**

5409

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	

5410 **Operation Output Parameters:**

5411

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification [ ]	return value	instance [ ]	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	
EndOfSequence	boolean	EndOfSequence	boolean	

5412 **Optional behavior:**

5413   •   CIM-XML allows implementations to optimize by not including properties in the returned
5414   instances that have a value of NULL.

5415 **Deviations:** None

5416 **C.1.21 CloseEnumeration**

5417 **CIM-XML Operation Name:** CloseEnumeration

5418 **Purpose:** Close an open enumeration session.

5419 **Operation Input Parameters:**

5420

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	

5421 **Operation Output Parameters:** None

5422 **Optional behavior:** None

5423 **Deviations:** None

5424 **C.1.22 EnumerationCount**

5425 **CIM-XML Operation Name:** EnumerationCount

5426 **Purpose:** Estimate the total number of remaining items in an open enumeration session.

5427 **Operation Input Parameters:**

5428

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
EnumerationContext	EnumerationContext	EnumerationContext	enumerationContext	

5429 **Operation Output Parameters:**

5430

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
EnumerationCount	uint64	return value	uint64	

5431 **Optional behavior:** None

5432 **Deviations:** None

5433 **C.1.23 InvokeMethod**

5434 **CIM-XML Operation Name:** The generic operation *InvokeMethod* corresponds to CIM-XML extrinsic
5435 method invocation on an instance. CIM-XML extrinsic method invocation on a class is covered by the
5436 generic operation *InvokeStaticMethod* (see C.1.24).

5437 **Purpose:** Invoke a method on an instance.

5438 **Operation Input Parameters:**

5439 This document does not define an operation name or parameters for extrinsic method invocation.
5440 DSP0201 defines the input and output parameters for extrinsic method invocation by means of the
5441 attributes and child elements of the XML elements METHODCALL and METHODRESPONSE. The table
5442 below therefore uses the names of these attributes and child elements in the mapping to generic
5443 operation parameters.

5444

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	

5445 1) The CIM-XML element *LOCALINSTANCEPATH* includes the model path portion of the instance path
5446   of the instance. The generic parameter *InstancePath* corresponds to the combination of the CIM-
5447   XML element *LOCALINSTANCEPATH* and the target namespace of the CIM-XML operation.

5448 **Operation Output Parameters:**

5449

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	

5450 **Optional behavior:** None

5451 **Deviations:** None

5452 **C.1.24 InvokeStaticMethod**

5453 **CIM-XML Operation Name:** The generic operation *InvokeStaticMethod* corresponds to CIM-XML
5454 extrinsic method invocation on a class. CIM-XML extrinsic method invocation on an instance is covered
5455 by the generic operation *InvokeMethod* (see C.1.23).

5456 **Purpose:** Invoke a static method on a class.

5457 **Operation Input Parameters:**

5458 This document does not define an operation name or parameters for extrinsic method invocation.
5459 DSP0201 defines the input and output parameters for extrinsic method invocation by means of the
5460 attributes and child elements of the XML elements METHODCALL and METHODRESPONSE. The table
5461 below therefore uses the names of these attributes and child elements in the mapping to generic
5462 operation parameters.

5463

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	

5464 1) The CIM-XML element *LOCALCLASSPATH* includes the model path portion of the class path of the
5465   class. The generic parameter *ClassPath* corresponds to the combination of the CIM-XML element
5466   *LOCALCLASSPATH* and the target namespace of the CIM-XML operation.

5467 **Operation Output Parameters:**

5468

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	

5469 **Optional behavior:** None

5470 **Deviations:** None

5471 **C.1.25 GetClass**

5472 **CIM-XML Operation Name:** GetClass

5473 **Purpose:** Retrieve a class given its class path.

5474 **Operation Input Parameters:**

5475

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)

5476 1) The CIM-XML parameter *ClassName* specifies the class name. The generic parameter *ClassPath*
5477   corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of
5478   the CIM-XML operation.

5479   2)   The defined behavior of generic operation *GetClass* conforms to the behavior of CIM-XML operation
5480   *GetClass* with *LocalOnly=false.*

5481   **Operation Output Parameters:**

5482

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
Class	ClassSpecification-WithPath	return value	class	See 1)

5483   1)   The CIM-XML return value includes the class declaration, without any class path information. The
5484   generic parameter *Class* needs to contain the class path in addition to the class declaration. A CIM
5485   client side mapping layer can remember the class path provided in the generic input parameter
5486   *ClassPath*, and add that to the generic output parameter *Class.*

5487   **Optional behavior:** None

5488   **Deviations:** None

5489   **C.1.26   DeleteClass**

5490   **CIM-XML Operation Name:** DeleteClass

5491   **Purpose:** Delete a class given its class path.

5492   **Operation Input Parameters:**

5493

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)

5494   1)   The CIM-XML parameter *ClassName* specifies the class name. The generic parameter *ClassPath*
5495   corresponds to the combination of the CIM-XML parameter *ClassName* and the target namespace of
5496   the CIM-XML operation.

5497   2)   **EXPERIMENTAL:** The experimental generic parameter *DeleteDependents* indicates whether
5498   dependent classes and instances are to be deleted as well. [DSP0223](#) defines the generic parameter
5499   *DeleteDependents* as optional. CIM-XML does not support deleting dependent classes and
5500   instances.

5501   **Operation Output Parameters:** None

5502   **Deviations:** None

5503   **C.1.27   ModifyClass**

5504   **CIM-XML Operation Name:** ModifyClass

5505   **Purpose:** Modify a class given its class path.

5506 **Operation Input Parameters:**

5507

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	

5508 1) The CIM-XML parameter *ModifiedClass* includes the name of the class that is being modified, and
5509 the modified class declaration. The combination of the class name portion of the CIM-XML
5510 parameter *ModifiedClass* and the target namespace of the CIM-XML operation corresponds to the
5511 generic parameter *ClassPath*.

5512 **Operation Output Parameters:** None

5513 **Optional behavior:** None

5514 **Deviations:** None

5515 **C.1.28 CreateClass**

5516 **CIM-XML Operation Name: CreateClass**

5517 **Purpose:** Create a class.

5518 **Operation Input Parameters:**

5519

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
NewClass	ClassSpecification	NewClass	class	

5520 **Operation Output Parameters:** None

5521 **Optional behavior:** None

5522 **Deviations:** None

5523 **C.1.29 EnumerateClasses**

5524 **CIM-XML Operation Name:** EnumerateClasses with ClassName being NULL

5525 **Purpose:** Retrieve the top classes (i.e., classes that have no superclasses) of a given namespace. The
5526 retrieved classes include their class paths.

5527   **Operation Input Parameters:**

5528

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	
IncludeSubclasses	boolean	DeepInheritance	boolean	
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
N/A	N/A	ClassName	className	See 1)
N/A	N/A	LocalOnly	boolean	See 2)

5529   1)   The defined behavior of generic operation *EnumerateClasses* conforms to the behavior of CIM-XML
5530   operation *EnumerateClasses* with *ClassName=NULL*.

5531   2)   The defined behavior of generic operation *EnumerateClasses* conforms to the behavior of CIM-XML
5532   operation *EnumerateClasses* with *LocalOnly=false*.

5533   **Operation Output Parameters:**

5534

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassList	ClassSpecification-WithPath [ ]	return value	class [ ]	See 1)

5535   1)   The CIM-XML return value includes the set of class declarations including class names, but without a
5536   class path. The generic parameter *ClassList* needs to contain the class path in addition to the class
5537   declaration. A CIM client side mapping layer can construct the class paths from the class names and
5538   the CIM-XML target namespace.

5539   **Optional behavior:** None

5540   **Deviations:** None

5541   **C.1.30   EnumerateClassNames**

5542   **CIM-XML Operation Name:** EnumerateClassNames with ClassName being NULL

5543   **Purpose:** Retrieve the class paths of the top classes (i.e., classes that have no superclasses) of a given
5544   namespace.

5545   **Operation Input Parameters:**

5546

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)

5547 1) The defined behavior of generic operation *EnumerateClassNames* conforms to the behavior of CIM-
5548 XML operation *EnumerateClassNames* with *ClassName=NULL*.

5549 **Operation Output Parameters:**

5550

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	className [ ]	See 1)

5551 1) The CIM-XML return value includes the set of class names, but without a class path. The generic
5552 parameter *ClassPathList* needs to contain the class paths. A CIM client side mapping layer can
5553 construct the class paths from the class names and the CIM-XML target namespace.

5554 **Optional behavior:** None

5555 **Deviations:** None

5556
5557
5558
5559

5560 **C.1.31 AssociatorClasses**

5561 **CIM-XML Operation Name:** Associators with ObjectName being a class path

5562 **Purpose:** Retrieve the classes that are associated with a given source class. The retrieved classes
5563 include their class paths.

5564 **Operation Input Parameters:**

5565

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	
IncludeQualifiers	boolean	IncludeQualifiers	boolean	
IncludeClassOrigin	boolean	IncludeClassOrigin	boolean	
IncludedProperties	PropertyName [ ]	PropertyList	string [ ]	

5566 1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter
5567 *ObjectName* and the target namespace of the CIM-XML operation.

5568 The generic operation *AssociatorClasses* corresponds to the CIM-XML operation *Associators* when
5569 a class path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *Associators*
5570 with an instance path for its *ObjectName* parameter is covered by the generic operation *Associators*
5571 (see C.1.7).

5572 **Operation Output Parameters:**

5573

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassList	ClassSpecification-WithPath [ ]	return value	objectWithPath [ ]	

5574 **Optional behavior:** None

5575 **Deviations:** None

5576 **C.1.32 AssociatorClassPaths**

5577 **CIM-XML Operation Name:** AssociatorNames with ObjectName being a class path

5578 **Purpose:** Retrieve the class paths of the classes that are associated with a given source class.

5579 **Operation Input Parameters:**

5580

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	

5581 1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter
5582 *ObjectName* and the target namespace of the CIM-XML operation.

5583 The generic operation *AssociatorClassPaths* corresponds to the CIM-XML operation
5584 *AssociatorNames* when a class path is passed in for its *ObjectName* parameter. Using the CIM-XML
5585 operation *AssociatorNames* with an instance path for its *ObjectName* parameter is covered by the
5586 generic operation *AssociatorNames* (see C.1.8).

5587 **Operation Output Parameters:**

5588

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	objectPath [ ]	

5589 **Optional behavior:** None

5590 **Deviations:** None

5591 **C.1.33 ReferenceClasses**

5592 **CIM-XML Operation Name:** References with ObjectName being a class path

5593 **Purpose:** Retrieve the association classes that reference a given source class. The retrieved classes
5594 include their class paths.

5595 **Operation Input Parameters:**

5596

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 [ ]	

5597 1) The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter
5598 *ObjectName* and the target namespace of the CIM-XML operation.

5599 The generic operation *ReferenceClasses* corresponds to the CIM-XML operation *References* when a
5600 class path is passed in for its *ObjectName* parameter. Using the CIM-XML operation *References* with
5601 an instance path for its *ObjectName* parameter is covered by the generic operation *References* (see
5602 C.1.9).

5603 2) The CIM-XML operation *References* does not support a means to filter by class name or role name
5604 of the associated classes on the other ends of the associations referencing the source class. The
5605 generic operation *ReferenceClasses* does support such filtering through its parameters
5606 *AssociatedClassName* and *AssociatedRoleName.* Since the defined behavior of the CIM-XML
5607 operation will result in including association classes that these two parameters could filter out, a
5608 mapping layer on the CIM client side can implement the behavior defined by these two generic
5609 parameters by eliminating association classes if these filter parameters are used.

5610 **Operation Output Parameters:**

5611

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
InstanceList	InstanceSpecification-WithPath [ ]	return value	objectWithPath [ ]	

5612 **Optional behavior:** None

5613 **Deviations:** None

5614 **C.1.34 ReferenceClassPaths**

5615 **CIM-XML Operation Name:** ReferenceNames with ObjectName being a class path

5616 **Purpose:** Retrieve the class paths of the association classes that reference a given class.

5617   **Operation Input Parameters:**

5618

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)

5619   1)   The generic parameter *ClassPath* corresponds to the combination of the CIM-XML parameter
5620   *ObjectName* and the target namespace of the CIM-XML operation.

5621   The generic operation *ReferenceClassPaths* corresponds to the CIM-XML operation
5622   *ReferenceNames* when a class path is passed in for its *ObjectName* parameter. Using the CIM-XML
5623   operation *ReferenceNames* with an instance path for its *ObjectName* parameter is covered by the
5624   generic operation *ReferenceNames* (see C.1.10).

5625   2)   The CIM-XML operation *References* does not support a means to filter by class name or role name
5626   of the associated classes on the other ends of the associations referencing the source class. The
5627   generic operation *ReferenceClassPaths* does support such filtering through its parameters
5628   *AssociatedClassName* and *AssociatedRoleName*. Since the defined behavior of the CIM-XML
5629   operation will result in including association classes that these two parameters could filter out, a
5630   mapping layer on the CIM client side can implement the behavior defined by these two generic
5631   parameters by eliminating association classes if these filter parameters are used.

5632   **Operation Output Parameters:**

5633

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
ClassPathList	ClassPath [ ]	return value	objectPath [ ]	

5634   **Optional behavior:** None

5635   **Deviations:** None

5636   **C.1.35 GetQualifierType**

5637   **CIM-XML Operation Name:** GetQualifier

5638   **Purpose:** Retrieve a qualifier type given its qualifier type path.

5639   **Operation Input Parameters:**

5640

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)

5641 1) The CIM-XML parameter *QualifierName* specifies the name of the qualifier type. The generic
5642 parameter *QualifierTypePath* corresponds to the combination of the CIM-XML parameter
5643 *QualifierName* and the target namespace of the CIM-XML operation.

5644 **Operation Output Parameters:**

5645

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierType	QualifierType	return value	qualifierDecl	See 1)

5646 1) The CIM-XML return value includes the qualifier type declaration including the qualifier type name,
5647 but without the namespace path portion of the full qualifier type path. The generic parameter
5648 *QualifierType* needs to contain the full qualifier type path in addition to the qualifier type declaration.
5649 A CIM client side mapping layer can remember the qualifier type path provided in the generic input
5650 parameter *QualifierTypePath*, and add that to the generic output parameter *QualifierType*.

5651 **Optional behavior:** None

5652 **Deviations:** None

5653 **C.1.36 DeleteQualifierType**

5654 **CIM-XML Operation Name:** DeleteQualifier

5655 **Purpose:** Delete a qualifier type given its qualifier type path.

5656 **Operation Input Parameters:**

5657

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypePath	QualifierTypePath	target namespace	N/A	See 1)
		QualifierName	string	See 1)

5658 1) The CIM-XML parameter *QualifierName* specifies the name of the qualifier type, i.e. the model path
5659 portion of its qualifier type path. The generic parameter *QualifierTypePath* corresponds to the
5660 combination of the CIM-XML parameter *QualifierName* and the target namespace of the CIM-XML
5661 operation.

5662 **Operation Output Parameters:** None

5663 **Deviations:** None

5664 **C.1.37 ModifyQualifierType**

5665 **CIM-XML Operation Name:** SetQualifier

5666 **Purpose:** Modify a qualifier type given its qualifier type path.

5667   **Operation Input Parameters:**

5668

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	

5669   1)   The CIM-XML parameter *QualifierDeclaration* includes the name of the qualifier type that is modified,
5670   i.e. the model path portion of its qualifier type, and the modified qualifier type declaration. The
5671   combination of the name of the qualifier type within the CIM-XML parameter *QualifierDeclaration* and
5672   the target namespace of the CIM-XML operation corresponds to the generic parameter
5673   *QualifierTypePath*.

5674   **Operation Output Parameters:** None

5675   **Optional behavior:** None

5676   **Deviations:**

5677   •   The generic operation *ModifyQualifierType* is required to fail if invoked on a non-existing
5678   qualifier type. The CIM-XML operation *SetQualifier* creates the qualifier type in this case. This
5679   deviation covers only an error case. A CIM client side mapping layer can expose the generic
5680   operation behavior by first testing for the existence of the qualifier type using the CIM-XML
5681   operation *GetQualifier*, before modifying it.

5682   **C.1.38 CreateQualifierType**

5683   **CIM-XML Operation Name:** SetQualifier

5684   **Purpose:** Create a CIM qualifier type.

5685   **Operation Input Parameters:**

5686

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	

5687   1)   The generic parameter *NamespacePath* corresponds to the combination of the qualifier type name
5688   specified in the CIM-XML parameter *NewQualifierType* and the target namespace of the CIM-XML
5689   operation.

5690   **Operation Output Parameters:**

5691

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypePath	QualifierTypePath	return value	instanceName	

5692 **Optional behavior:** None

5693 **Deviations:**

5694 • The generic operation *CreateQualifierType* is required to fail if invoked on an existing qualifier
5695 type. The CIM-XML operation *SetQualifier* modifies the qualifier type in this case. This deviation
5696 covers only an error case. A CIM client side mapping layer can expose the generic operation
5697 behavior by first testing for the existence of the qualifier type using the CIM-XML operation
5698 *GetQualifier*, before creating it.

5699 **C.1.39 EnumerateQualifierTypes**

5700 **CIM-XML Operation Name:** EnumerateQualifiers

5701 **Purpose:** Retrieve the qualifier types of a given namespace. The retrieved qualifier types include their
5702 qualifier type paths.

5703 **Operation Input Parameters:**

5704

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
NamespacePath	NamespacePath	target namespace	N/A	

5705 **Operation Output Parameters:**

5706

Generic Name	Generic Type	CIM-XML Name	CIM-XML Type	Description
QualifierTypeList	QualifierTypeWithPath [ ]	return value	qualifierDecl [ ]	See 1)

5707 1) The CIM-XML return value includes the set of qualifier type declarations including their names, but
5708 without namespace paths. The generic parameter QualifierTypeList needs to contain the qualifier
5709 type paths in addition to the set of qualifier type declarations. A CIM client side mapping layer can
5710 construct the qualifier type paths from the qualifier names and the CIM-XML target namespace.

5711 **Optional behavior:** None

5712 **Deviations:** None

5713   **ANNEX D**
5714   (informative)
5715
5716
5717   **Change Log**

Version	Date	Description
1.0	1999-06-02	
1.1	2003-01-06	DMTF Standard
1.2	2007-01-09	DMTF Standard
1.3.0	2008-10-15	DMTF Standard
1.3.1	2009-07-29	DMTF Standard

Version	Date	Description
1.4.0	2013-08-26	DMTF Standard 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)* Deprecate the list agreed to by the WG Change all Generic Operation Names to match updated Generic Operations Specification.

5718

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