1

5 # CIM Operations over HTTP

9

10 Copyright Notice

11 Copyright © 2009 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

32

CONTENTS

145

146   **Tables**

155

156 # Foreword

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

158   Introduction

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

163   This document is structured as follows:

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

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

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

170   –   HTTP version support

171   –   Use of standard HTTP headers

172   –   HTTP error codes

173   –   Security considerations

174   **Requirements**

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

179   The following criteria are applied to the representation of CIM messages in XML using DSP0201 and
180   DSP0203:

181   • Each CIM message is completely described in XML; completeness is favored over conciseness.

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

186   • The set of CIM messages is classified into functional profiles to accommodate a range of
187   implementations varying from complete support of all messages to support of a minimal subset.
188   The number of functional profiles is kept as small as possible to encourage interoperability, and
189   mechanisms provided by different CIM implementations can declare their level of support.

190   The following criteria are applied to the HTTP encapsulation of CIM Messages herein:

191   • In recognition of the large installed base of HTTP/1.0 systems, the encapsulation is designed to
192   support both HTTP/1.0 and HTTP/1.1.

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

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

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

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

205   • There is a clear and unambiguous encapsulation of the CIM message payload within the HTTP
206   message. Conciseness of the encapsulation is of secondary importance.

207   **CIM Operations over HTTP**

1   Scope

209   The Common Information Model (CIM) ([DSP0004](#)) is an object-oriented information model defined by the
210   Distributed Management Task Force (DMTF) that provides a conceptual framework for describing
211   management data.

212   The Hypertext Transfer Protocol (HTTP) ([RFC 1945](#), [RFC 2068](#), [RFC 2616](#)) is an application-level
213   protocol for distributed, collaborative, hypermedia information systems. This generic stateless protocol
214   can be used for many tasks through extension of its request methods, error codes, and headers.

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

218   This document defines a mapping of CIM messages onto HTTP that allows implementations of CIM to
219   interoperate in an open, standardized manner. It is based on the *CIM XML DTD* ([DSP0201](#) and
220   [DSP0203](#)) that defines the XML schema for CIM objects and messages.

2   Normative References

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

225   DMTF DSP0004, *Common Information Model (CIM) Infrastructure 2.5*,
226   http://www.dmtf.org/standards/published_documents/DSP0004_2.5.pdf

227   DMTF DSP0201, *Specification for the Representation of CIM in XML 2.3*,
228   http://www.dmtf.org/standards/published_documents/DSP0201_2.3.pdf

229   DMTF DSP0203, *CIM XML DTD 2.3*,
230   http://www.dmtf.org/standards/published_documents/DSP0203_2.3.dtd

231   IETF RFC 1766, *Tags for the Identification of Languages*, March 1995, http://www.ietf.org/rfc/rfc1766.txt

232   IETF RFC 1945, *Hypertext Transfer Protocol – HTTP/1.0*, May 1996, http://www.ietf.org/rfc/rfc1945.txt

233   IETF RFC 2068, *Hypertext Transfer Protocol – HTTP/1.1*, January 1997, http://www.ietf.org/rfc/rfc2068.txt

234   IETF RFC 2069, *An Extension to HTTP: Digest Access Authentication*, January 1997,
235   http://www.ietf.org/rfc/rfc2069.txt

236   IETF RFC 2277, *IETF Policy on Character Sets and Languages*, January 1998,
237   http://www.ietf.org/rfc/rfc2277.txt

238   IETF RFC 2279, *UTF-8, a transformation format of Unicode and ISO 10646*, January 1998,
239   http://www.ietf.org/rfc/rfc2279.txt

240   IETF RFC 2376, *XML Media Types*, July 1998, http://www.ietf.org/rfc/rfc2376.txt

241   IETF RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*, August 1998,
242   http://www.ietf.org/rfc/rfc2396.txt

243 IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999, http://www.ietf.org/rfc/rfc2616.txt

244 IETF RFC 2617, *HTTP Authentication: Basic and Digest Access Authentication*, June 1999,
245 http://www.ietf.org/rfc/rfc2617.txt

246 IETF RFC 2774, *HTTP Extension Framework*, February 2000, http://www.ietf.org/rfc/rfc2774.txt

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

249 W3C Recommendation, *Namespaces in XML*, Jan 1999, http://www.w3.org/TR/1999/REC-xml-names-
250 19990114/

251 W3C, *XML Schema Part 1: Structures*, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-
252 20010502/

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

254   2.1   Other References

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

257   3   Terms and Definitions

258 Throughout this document, the following terms and definitions apply. This specification uses the same
259 notational conventions and basic parsing constructs that are defined in RFC 2068.

260 **3.1**
261 **can**
262 used for statements of possibility and capability, whether material, physical, or causal

263 **3.2**
264 **cannot**
265 used for statements of possibility and capability, whether material, physical, or causal

266 **3.3**
267 **conditional**
268 indicates requirements that must be strictly followed to conform to the document when the specified
269 conditions are met

270 **3.4**
271 **mandatory**
272 indicates requirements that must be strictly followed to conform to the document, with no permitted
273 deviations

274 **3.5**
275 **may**
276 indicates a course of action permissible within the limits of the document

277 **3.6**
278 **need not**
279 indicates a course of action permissible within the limits of the document

280 **3.7**
281 **optional**
282 indicates a course of action permissible within the limits of the document

283 **3.8**
284 **shall**
285 indicates requirements that must be strictly followed to conform to the document, with no permitted
286 deviations

287 **3.9**
288 **shall not**
289 indicates requirements that must be strictly followed to conform to the document, with no permitted
290 deviations

291 **3.10**
292 **should**
293 indicates that among several possibilities, one is recommended as particularly suitable, without
294 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required

295 **3.11**
296 **should not**
297 indicates that a certain possibility or course of action is deprecated but not prohibited

298 **3.12**
299 **unspecified**
300 indicates that this profile does not define any constraints for the referenced CIM element or operation

301 **3.13**
302 **CIM Message**
303 a well-defined request or response data packet used to exchange information among CIM products

304 **3.14**
305 **CIM Operation Message**
306 a CIM message used to invoke an operation on the target namespace

307 **3.15**
308 **CIM Export Message**
309 a CIM message used to communicate information about a CIM namespace or element that is foreign to
310 the target

311 **3.16**
312 **Operation Request Message**
313 an XML document that is loosely valid with respect to the CIM XML DTD and that contains a
314 <MESSAGE> subelement under the root <CIM> node that has a <MULTIREQ> or <SIMPLEREQ>
315 subelement under this subelement

316 4   Abbreviated Terms and Document Conventions

317 4.1   Abbreviated Terms

318   The following symbols and abbreviations are used in this document.

319 4.1.1
320   **CIM**
321   Common Information Model

322 4.1.2
323   **DTD**
324   document type definition

325 4.1.3
326   **HTTP**
327   Hypertext Transfer Protocol

328 4.1.4
329   **XML**
330   Extensible Markup Language

331 4.2   Document Conventions

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

333   • The "**DEPRECATION NOTE:**" label preceding a paragraph indicates that the paragraph explains a
334   deprecated element.

335   • The "**DEPRECATED.**" label before a list item indicates that the information in that list item is
336   deprecated.

337   • The "(DEPRECATED)" label after a heading applies to the entire clause for that heading.

338   • The "(DEPRECATED)" label at the end of a line in a code fragment or an example indicates that the
339   particular line of the code fragment or example is deprecated.

340 5   CIM Message Syntax and Semantics

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

344   • *CIM operation message*. This type of message is used to invoke an operation on the target
345   namespace.

346   • *CIM export message.* This type of message is used to communicate information about a CIM
347   namespace or element that is foreign to the target. It is informational only and does not define
348   an operation on the target CIM namespace or even imply the existence of a target namespace.

349   This clause describes the syntax and semantics of CIM messages independently of their encapsulation
350   within a particular protocol such as HTTP. XML is used as the basis for this description, and in particular
351   the CIM XML DTD (DSP0201 and DSP0203).

5.1   Well-Formed, Valid, and Loosely Valid Documents

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

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

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

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

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

- A CIM client may include a DOCTYPE element in a CIM message request. If so, an external declaration should be used. In-lining of the complete DTD within a message is discouraged.

- A CIM client may elect to validate a received CIM message response.

- If a CIM client elects not to validate a received CIM message, then loose validation shall be enforced.

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

5.2   Operational Semantics

The CIM XML DTD (DSP0201 and DSP0203) defines a subelement under the root <CIM> element called <MESSAGE>, which contains one of the following subelements:

- CIM operation message subelements
    - <SIMPLEREQ>
    - <SIMPLERSP>
    - <MULTIREQ>
    - <MULTIRSP>
- CIM export message subelements
    - <SIMPLEXPREQ>
    - <SIMPLEXPRSP>
    - <MULTIEXPREQ>
    - <MULTIEXPRSP>

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

396 • *Operation request message.* Contains under the root <CIM> node a <MESSAGE> subelement
397   that has a <MULTIREQ> or <SIMPLEREQ> subelement under it.

398 • *Operation response message.* Contains under the root <CIM> node a <MESSAGE>
399   subelement that has a <MULTIRSP> or <SIMPLERSP> subelement under it.

400 • *Export request message.* Contains under the root <CIM> node a <MESSAGE> subelement that
401   has a <MULTIEXPREQ> or <SIMPLEEXPREQ> subelement under it.

402 • *Export response message.* Contains under the root <CIM> node a <MESSAGE> subelement
403   that has a <MULTIEXPRSP> or <SIMPLEEXPRSP> subelement under it.

404 The phrase *CIM message request* refers to either an operation request message or an export request
405 message. The phrase *CIM message response* refers to either an operation response message or an
406 export response message.

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

410 Any CIM message conforming to this specification shall have a minimum value of "1.0" and a maximum
411 value that is equal to the latest version of this specification for the PROTOCOLVERSION attribute of the
412 <MESSAGE> element.

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

416   – A <MULTIRSP> subelement, if the operation request message contains a <MULTIREQ>
417   subelement.

418   – A <SIMPLERSP> subelement, if the operation request message contains a
419   <SIMPLEREQ> subelement.

420 A *simple operation request* is an operation request message that contains a <SIMPLEREQ> subelement.
421 A *simple operation response* is an Operation Response Message that contains a <SIMPLERSP>
422 subelement.

423 A *multiple operation request* is an operation request message that contains a <MULTIREQ> subelement.
424 A *multiple operation response* is an operation response message that contains a <MULTIRSP>
425 subelement.

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

429   – A <MULTIEXPRSP> subelement if the export request message contained a
430   <MULTIEXPREQ> subelement, or

431   – A <SIMPLEEXPRSP> subelement if the export request message contained a
432   <SIMPLEEXPREQ> subelement.

433 A simple export request is an export request message that contains a <SIMPLEEXPREQ> subelement. A
434 simple export response is an export response message that contains a <SIMPLEEXPRSP> subelement.

435 A multiple export request is an export request message that contains a <MULTIEXPREQ> subelement. A
436 multiple export response is an export response message that contains a <MULTIEXPRSP> subelement.

5.3   CIM Operation Syntax and Semantics

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

5.3.1   Method Invocations

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

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

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

In addition, intrinsic methods are made against a CIM namespace. Extrinsic methods are invoked on a CIM class (if static) or instance otherwise. Intrinsic methods are defined in 5.3.2.

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

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

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

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

- Input parameters may be supplied in any order.

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

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

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

- Output parameters may be supplied in any order.

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

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

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

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

5.3.1.1   Simple Operations

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

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

482 • A <LOCALCLASSPATH> subelement identifying the CIM class on which the method is to be
483   invoked if the method is static.

484 • A <LOCALINSTANCEPATH> subelement identifying the CIM instance on which the method is
485   otherwise to be invoked.

5.3.1.2   Multiple Operations

486

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

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

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

496 Multiple operations conveniently allow multiple method invocations to be batched into a single HTTP
497 message. Batching reduces the number of roundtrips between a CIM client and a CIM server and allows
498 the CIM server to make internal optimizations if it chooses. Note that multiple operations do not confer
499 any transactional capabilities in processing the request. For example, the CIM server does not have to
500 guarantee that the constituent method calls either all fail or succeed, only that the entity make a "best
501 effort" to process the operation. However, servers shall finish processing each operation in a batched
502 operation before executing the next one. Clients shall recognize that the order of operations within a
503 batched operation is significant.

504 Not all CIM servers support multiple operations; the way they declare support for this feature is defined in
505 7.5.

5.3.1.3   Status Codes

506

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

510 • a mandatory status code

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

512 • zero or more CIM_Error instances

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

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

521 Not all methods are expected to return all the status codes listed in Table 1. For <u>intrinsic methods</u>, the
522 relevant clause on each method in this specification defines the error codes expected to be returned. For
523 extrinsic methods, 5.3.5 specifies which of the codes in Table 1 can be used.

524   **Table 1 – Status Codes Returned by an <Error> Subelement**

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 CIM 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 CIM 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.
CIM_ERR_NAMESPACE_NOT_EMPTY	20	The specified namespace is not empty.

Symbolic Name	Code	Definition
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 CIM 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 filter in the enumeration is not supported by the CIM server.
CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED	26	The CIM server does not support continuation on error.
CIM_ERR_SERVER_LIMITS_EXCEEDED	27	The CIM server has failed the operation based upon exceeding server limits.
CIM_ERR_SERVER_IS_SHUTTING_DOWN	28	The CIM server is shutting down and cannot process the operation.

5.3.2   Intrinsic Methods

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

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

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

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

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

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

541 This notation uses the NULL qualifier for parameters whose values can be specified as NULL in a method
542 call. A NULL (unassigned) value for a parameter is specified by an <IPARAMVALUE> or
543 <PARAMVALUE> element with no subelement. For parameters without the NULL qualifier, the CIM client
544 shall specify a value by including a suitable subelement for the <IPARAMVALUE> or <PARAMVALUE>
545 element.

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

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

550   • Simple values use the <VALUE> subelement within an <IPARAMETER> element for method
551   parameters or within an <IRETURNVALUE> element for method return values.

552   • Array values use the <VALUE.ARRAY> subelement within an <IPARAMETER> element for
553   method parameters or within an <IRETURNVALUE> element for method return values.

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

557   **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
<enumerationContext>	ENUMERATIONCONTEXT

558 **5.3.2.1 GetClass**

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

```
560      GetClass
561      <class>GetClass (
562          [IN] <className> ClassName,
563          [IN,OPTIONAL] boolean LocalOnly = true,
564          [IN,OPTIONAL] boolean IncludeQualifiers = true,
565          [IN,OPTIONAL] boolean IncludeClassOrigin = false,
566          [IN,OPTIONAL,NULL] string PropertyList [] = NULL
567      )
```

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

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

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

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

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

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

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

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

595 • CIM_ERR_ACCESS_DENIED

596 • CIM_ERR_INVALID_NAMESPACE

597 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise
598 incorrect parameters)

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

600 • CIM_ERR_FAILED (Some other unspecified error occurred.)

601 **5.3.2.2 GetInstance**

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

603   **GetInstance**
604   <u><instance></u>**GetInstance** (
605   [IN] <u><instanceName></u> InstanceName,
606   [IN,OPTIONAL] boolean LocalOnly = true, (DEPRECATED)
607   [IN,OPTIONAL] boolean IncludeQualifiers = false, (DEPRECATED)
608   [IN,OPTIONAL] boolean IncludeClassOrigin = false,
609   [IN,OPTIONAL,NULL] string PropertyList [] = NULL
610   )

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

612 **DEPRECATION NOTE:** With the 1.2 release of this specification, the LocalOnly parameter is
613 DEPRECATED. LocalOnly filtering, as defined in 1.1, will not be supported in the next major revision of
614 this specification. In the 1.1 version of this specification, the definition of the LocalOnly parameter was
615 incorrectly modified. This change introduced a number of interoperability and backward compatibility

616   problems for CIM clients using the LocalOnly parameter to filter the set of properties returned. The DMTF
617   strongly recommends that CIM clients set LocalOnly to `false` and do not use this parameter to filter the
618   set of properties returned. To minimize the impact of this recommendation on CIM clients, a CIM server
619   may choose to treat the value of the LocalOnly parameter as `false` for all requests. A CIM server shall
620   consistently support a single interpretation of the LocalOnly parameter. Refer to ANNEX B for additional
621   details.

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

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

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

641   If the `PropertyList` contains duplicate elements, the server shall ignore the duplicates but otherwise
642   process the request normally. If the `PropertyList` contains elements that are invalid Property names
643   for the target instance, the server shall ignore them but otherwise process the request normally.

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

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

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

654   •   CIM_ERR_ACCESS_DENIED

655   •   CIM_ERR_INVALID_NAMESPACE

656   •   CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
657   incorrect parameters)

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

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

661 • CIM_ERR_FAILED (some other unspecified error occurred)

5.3.2.3 DeleteClass

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

```
664        DeleteClass
665        void  DeleteClass (
666            [IN] <className> ClassName
667        )
```

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

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

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

674 • CIM_ERR_ACCESS_DENIED

675 • CIM_ERR_NOT_SUPPORTED

676 • CIM_ERR_INVALID_NAMESPACE

677 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
678   incorrect parameters)

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

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

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

684 • CIM_ERR_FAILED (Some other unspecified error occurred.)

5.3.2.4 DeleteInstance

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

```
687        DeleteInstance
688        void  DeleteInstance (
689            [IN] <instanceName> InstanceName
690        )
```

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

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

696 If DeleteInstance is successful, the CIM server removes the specified instance.

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

700 • CIM_ERR_ACCESS_DENIED

701 • CIM_ERR_NOT_SUPPORTED (by the CIM server for this operation)

702 • CIM_ERR_INVALID_NAMESPACE

703 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
704   incorrect parameters)

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

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

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

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

5.3.2.5   CreateClass

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

```
715        CreateClass
716        void CreateClass (
717            [IN] <class> NewClass
718        )
```

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

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

722 • The server shall ignore any `CLASSORIGIN` and `PROPAGATED` attributes in the `NewClass`.

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

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

729   – Any properties, methods, or qualifiers in the subclass not defined in the superclass are
730   created as new elements of the subclass. In particular, the server shall set the
731   `CLASSORIGIN` attribute on the new properties and methods to the name of the subclass
732   and ensure that all others preserve their `CLASSORIGIN` attribute value from that defined in
733   the superclass.

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

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

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

742   – Any qualifier propagated from the superclass cannot be modified in the subclass if the
743   `OVERRIDABLE` attribute of that qualifier is set to `false` in the superclass. It is a client error

744   to specify such a qualifier in the `NewClass` with a definition different than that in the
745   superclass (where definition encompasses the name, type, and flavor attribute settings of
746   the `<QUALIFIER>` element and the value of the qualifier).

747   If CreateClass is successful, the CIM server creates the specified class.

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

751   • CIM_ERR_ACCESS_DENIED

752   • CIM_ERR_NOT_SUPPORTED

753   • CIM_ERR_INVALID_NAMESPACE

754   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
755   incorrect parameters)

756   • CIM_ERR_ALREADY_EXISTS (The CIM class already exists.)

757   • CIM_ERR_INVALID_SUPERCLASS (The putative CIM class declares a non-existent
758   superclass.)

759   • CIM_ERR_FAILED (Some other unspecified error occurred.)

5.3.2.6   CreateInstance

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

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

767   **DEPRECATION NOTE:** The use of qualifiers on instances is DEPRECATED and may be removed in a
768   future version of this specification. A CIM client cannot rely on any qualifiers included in the NewInstance
769   to have any impact on the operation. It is recommended that the CIM server ignore any qualifiers included
770   in the instance. The `NewInstance` input parameter defines the new instance. The proposed definition
771   shall be a correct instance definition for the underlying CIM class according to DSP0004.

772   In creating the new instance, the CIM server shall conform to the following rules and ensure that they are
773   applied:

774   • The server shall ignore any `CLASSORIGIN` and `PROPAGATED` attributes in the `NewInstance`.

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

777   • All properties of the instance preserve their `CLASSORIGIN` attribute value from that defined in
778   the class.

779   • If a property is specified in the `NewInstance` parameter, the value assigned to that property in
780   the instance (including NULL) becomes the value of the property for the instance. Note that it is
781   a client error to specify a property that does not belong to the class.

782   • If a property of the class is not specified in the instance, then the instance inherits that property
783   without modification.

784   • **DEPRECATION NOTE:** Use of the TOINSTANCE attribute is DEPRECATED. Servers may
785   choose to ignore TOINSTANCE. Servers that do not ignore TOINSTANCE shall interpret it so

786   that any qualifiers defined in the class with a TOINSTANCE attribute value of `true` appear in
787   the instance. Qualifiers in the class with a value of `false` shall not be propagated to the
788   instance.

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

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

798   If CreateInstance is unsuccessful, this method shall return one of the following status codes, where the
799   error 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_NOT_SUPPORTED (by the CIM server for this operation)

803   •   CIM_ERR_INVALID_NAMESPACE

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

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

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

809   •   CIM_ERR_ALREADY_EXISTS (The CIM instance already exists.)

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

812   **5.3.2.7   ModifyClass**

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

815   **ModifyClass**
816   void **ModifyClass** (
817   [IN] <class> ModifiedClass
818   )

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

821   In modifying the class, the CIM server shall conform to the following rules:

822   •   The CIM server shall ignore any `CLASSORIGIN` and `PROPAGATED` attributes in the
823   `ModifiedClass`.

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

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

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

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

834   –   Any properties, methods, or qualifiers in the subclass not defined in the superclass are
835   created as elements of the subclass. In particular, the server shall set the `CLASSORIGIN`
836   attribute on the new properties and methods to the name of the subclass and shall ensure
837   that all other others preserve their `CLASSORIGIN` attribute value from that defined in the
838   superclass.

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

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

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

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

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

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

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

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

863   •   CIM_ERR_ACCESS_DENIED

864   •   CIM_ERR_NOT_SUPPORTED

865   •   CIM_ERR_INVALID_NAMESPACE

866   •   CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
867   incorrect parameters)

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

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

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

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

875   • CIM_ERR_FAILED (Some other unspecified error occurred.)

876   **5.3.2.8   ModifyInstance**

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

879   **ModifyInstance**
880   void **ModifyInstance** (
881   [IN] <namedInstance> ModifiedInstance,
882   [IN, OPTIONAL] boolean IncludeQualifiers = true,   (DEPRECATED)
883   [IN, OPTIONAL, NULL] string propertyList[] = NULL
884   )

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

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

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

895   If the PropertyList input parameter is not NULL, the members of the array define one or more property
896   names. Only properties specified in the PropertyList are modified. Properties of the ModifiedInstance that
897   are missing from the PropertyList are ignored. If the PropertyList is an empty array, no properties are
898   modified. If the PropertyList is NULL, the set of properties to be modified consists of those of
899   ModifiedInstance with values different from the current values in the instance to be modified.

900   If the PropertyList contains duplicate elements, the server shall ignore them but otherwise process the
901   request normally. If the PropertyList contains elements that are invalid property names for the instance to
902   be modified, the server shall reject the request. If a property to be modified as previously defined cannot
903   be modified because it is a key property, non-writable, or cannot be modified at this time for any other
904   reason, the server shall reject the request. Non-writable properties are those defined to be non-writable in
905   the schema implemented for the creation class of the instance to be modified. The value of the WRITE
906   qualifier as defined in the schema for the creation class (or any of its superclasses) has no effect on the
907   behavior of ModifyInstance.

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

909   • The server shall ignore any CLASSORIGIN and PROPAGATED attributes in the
910   ModifiedInstance.

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

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

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

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

919   • If a property is to be modified as previously defined, its value in the ModifiedInstance (including
920   NULL) becomes its new value in the instance to be modified.

921   • If a property is not specified in the ModifiedInstance but is specified in the PropertyList, then the
922   class-defined default value (or NULL if none is defined) becomes its new value in the instance
923   to be modified.

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

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

935   If ModifyInstance is successful, all properties to be modified are updated in the specified instance.

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

939   • CIM_ERR_ACCESS_DENIED

940   • CIM_ERR_NOT_SUPPORTED (by the CIM server for this operation)

941   • CIM_ERR_INVALID_NAMESPACE

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

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

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

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

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

5.3.2.9   EnumerateClasses

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

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

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

962   If the DeepInheritance input parameter is true, all subclasses of the specified class should be
963   returned. If the ClassName input parameter is absent, this implies that all classes in the target
964   namespace should be returned. If DeepInheritance is false, only immediate child subclasses are
965   returned. If the ClassName input parameter is NULL, this implies that all base classes in the target
966   namespace should be returned. This definition of DeepInheritance applies only to the
967   EnumerateClasses and EnumerateClassName operations.

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

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

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

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

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

982   • CIM_ERR_ACCESS_DENIED

983   • CIM_ERR_NOT_SUPPORTED

984   • CIM_ERR_INVALID_NAMESPACE

985   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
986   incorrect parameters)

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

988   • CIM_ERR_FAILED (Some other unspecified error occurred.)

989   **5.3.2.10 EnumerateClassNames**

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

992   **EnumerateClassNames**
993   <className>***EnumerateClassNames** (
994   [IN,OPTIONAL,NULL] <className> ClassName = NULL,
995   [IN,OPTIONAL] boolean DeepInheritance = false
996   )

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

998   If the DeepInheritance input parameter is true, the names of all subclasses of the specified class
999   should be returned. If the ClassName input parameter is absent, this implies that the names of all classes
1000   in the target namespace should be returned. If DeepInheritance is false, only the names of
1001   immediate child subclasses are returned. If the ClassName input parameter is NULL, this implies that the

1002   names of all base classes in the target namespace should be returned. This definition of
1003   `DeepInheritance` applies only to the EnumerateClasses and EnumerateClassName operations.

1004   If EnumerateClassNames is successful, the method returns zero or more names of classes that meet the
1005   requested criteria.

1006   If EnumerateClassNames is unsuccessful, this method returns one of the following status codes, where
1007   the error returned is the first applicable error in the list, starting with the first element and working down.
1008   Any additional method-specific interpretation of the error is enclosed in parentheses.

1009   •   CIM_ERR_ACCESS_DENIED

1010   •   CIM_ERR_NOT_SUPPORTED

1011   •   CIM_ERR_INVALID_NAMESPACE

1012   •   CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1013   incorrect parameters)

1014   •   CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not
1015   exist.)

1016   •   CIM_ERR_FAILED (Some other unspecified error occurred.)

1017   **5.3.2.11 EnumerateInstances**

1018   The EnumerateInstances operation enumerates instances of a CIM class in the target namespace,
1019   including instances in the class and any subclasses in accordance with the polymorphic nature of CIM
1020   objects:

1021   **EnumerateInstances**
1022   <namedInstance>***EnumerateInstances** (
1023   [IN] <className> ClassName,
1024   [IN,OPTIONAL] boolean LocalOnly = true, (DEPRECATED)
1025   [IN,OPTIONAL] boolean DeepInheritance = true,
1026   [IN,OPTIONAL] boolean IncludeQualifiers = false, (DEPRECATED)
1027   [IN,OPTIONAL] boolean IncludeClassOrigin = false,
1028   [IN,OPTIONAL,NULL] string PropertyList [] = NULL
1029   )

1030   The `ClassName` input parameter defines the class that is the basis for the enumeration.

1031   **DEPRECATION NOTE:** With the 1.2 release of this specification, the `LocalOnly` parameter is
1032   DEPRECATED. LocalOnly filtering, as defined in 1.1, will not be supported in the next major revision of
1033   this specification. In the 1.1 version of this specification, the definition of the `LocalOnly` parameter was
1034   incorrectly modified. This change introduced a number of interoperability and backward compatibility
1035   problems for CIM clients using the `LocalOnly` parameter to filter the set of properties returned. The
1036   DMTF strongly recommends that CIM clients set `LocalOnly` to `false` and do not use this parameter to
1037   filter the set of properties returned. To minimize the impact of this recommendation on CIM clients, a CIM
1038   server may choose to treat the value of the `LocalOnly` parameter as `false` for all requests. A CIM
1039   server shall consistently support a single interpretation of the `LocalOnly` parameter. Refer to ANNEX B
1040   for details.

1041   If the `DeepInheritance` input parameter is `false`, each returned instance shall not include any
1042   properties added by subclasses of the specified class. If it is `true`, no additional filtering is defined.

1043   **DEPRECATION NOTE:** The use of the `IncludeQualifiers` parameter is DEPRECATED and it may
1044   be removed in a future version of this specification. The definition of `IncludeQualifiers` is ambiguous
1045   and when this parameter is set to `true`, CIM clients cannot be assured that any qualifiers will be

1046   returned. A CIM client should always set this parameter to `false`. To minimize the impact of this
1047   recommendation on CIM clients, a CIM server may choose to treat the value of `IncludeQualifiers`
1048   as `false` for all requests. The preferred behavior is to use the class operations to receive qualifier
1049   information and not depend on any qualifiers in this response. If the `IncludeQualifiers` input
1050   parameter is `true`, all qualifiers for the instance, (including qualifiers on the instance and on any returned
1051   properties, shall be included as `<QUALIFIER>` elements in the response. If it is `false`, no
1052   `<QUALIFIER>` elements are present in the returned instance.

1053   If the `IncludeClassOrigin` input parameter is `true`, the CLASSORIGIN attribute shall be present on
1054   all appropriate elements in each returned Instance. If it is `false`, no CLASSORIGIN attributes are
1055   present.

1056   If the `PropertyList` input parameter is not NULL, the members of the array define one or more
1057   property names of the designated class. This definition may include inherited property names or property
1058   names explicitly defined in the designated class. However, it may not include property names added in
1059   subclasses of the designated class. Each returned instance shall not include elements for any properties
1060   missing from this list. Note that `PropertyList` acts as an additional filter on the properties defined by
1061   the `LocalOnly` and `DeepInheritance` input parameters; if the PropertyList includes a property that is
1062   not in the set defined by the `LocalOnly` and `DeepInheritance` combination, the element for the
1063   property shall not be returned. If `PropertyList` is an empty array, no elements for properties are
1064   included in the returned instances. If `PropertyList` is NULL, no additional filtering is defined.

1065   If the `PropertyList` contains duplicate elements, the server shall ignore the duplicates but otherwise
1066   process the request normally. If the `PropertyList` contains elements which are invalid Property names
1067   for any target instance, the server shall ignore such entries but otherwise process the request normally.

1068   Properties with the NULL value may be omitted from the response, even if the CIM client has not
1069   requested the exclusion of the property through the `LocalOnly`, `DeepInheritance`, or `PropertyList`
1070   filters. The CIM client shall interpret such omitted properties as NULL. Note that the CIM client cannot
1071   make any assumptions about properties omitted as a result of using any `LocalOnly`,
1072   `DeepInheritance`, or `PropertyList` filters.

1073   If EnumerateInstances is successful, the method returns zero or more named instances that meet the
1074   required criteria. These instances shall have all properties defined in and inherited by their respective
1075   classes, reduced by any properties excluded as a result of using the `LocalOnly`, `DeepInheritance`, or
1076   `PropertyList` filters and further reduced by any NULL-valued properties omitted from the response.

1077   If EnumerateInstances is unsuccessful, this method shall return one of the following status codes, where
1078   the error returned is the first applicable error in the list, starting with the first element and working down.
1079   Any additional method-specific interpretation of the error is enclosed in parentheses.

1080   • CIM_ERR_ACCESS_DENIED

1081   • CIM_ERR_NOT_SUPPORTED (by the CIM server for this operation)

1082   • CIM_ERR_INVALID_NAMESPACE

1083   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1084   incorrect parameters)

1085   • CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not
1086   exist.)

1087   • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the specified class and all
1088   of its subclasses, if provided.)

1089   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1090 **5.3.2.12 EnumerateInstanceNames**

1091 The EnumerateInstanceNames operation enumerates the names (model paths) of the instances of a CIM
1092 class in the target namespace, including instances in the class and any subclasses in accordance with
1093 the polymorphic nature of CIM objects:

```
1094        EnumerateInstanceNames
1095        <instanceName>*EnumerateInstanceNames (
1096            [IN] <className> ClassName
1097        )
```

1098 The ClassName input parameter defines the class that is the basis for the enumeration.

1099 If EnumerateInstanceNames is successful, the method returns zero or more InstanceNames (referred to
1100 in DSP0004 as a model path) that meet the requested criteria. The InstanceName shall specify the
1101 class from which the instance is instantiated, not any of its base classes. Note that this class may be
1102 different from the class specified as input.

1103 If EnumerateInstanceNames is unsuccessful, this method shall return one of the following status codes,
1104 where the error returned is the first applicable error in the list, starting with the first element and working
1105 down. Any additional method-specific interpretation of the error is enclosed in parentheses.

1106 • CIM_ERR_ACCESS_DENIED

1107 • CIM_ERR_NOT_SUPPORTED (by the CIM server for this operation)

1108 • CIM_ERR_INVALID_NAMESPACE

1109 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1110   incorrect parameters)

1111 • CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not
1112   exist.)

1113 • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the specified class and all
1114   of its subclasses, if provided.)

1115 • CIM_ERR_FAILED (Some other unspecified error occurred.)

1116 **5.3.2.13 ExecQuery**

1117 The ExecQuery operation executes a query against the target namespace:

```
1118        ExecQuery
1119        <object>*ExecQuery (
1120            [IN] string QueryLanguage,
1121            [IN] string Query
1122        )
```

1123 The QueryLanguage input parameter defines the query language in which the query parameter is
1124 expressed.

1125 The Query input parameter defines the query to be executed. The results of the query shall be
1126 constrained to contain only CIM classes that exist in the target namespace or CIM instances whose
1127 classes exist in the target namespace. Note that any instances in the result set may or may not exist in
1128 any namespace. Note that for query languages supporting select-lists and from-clauses, this implies that
1129 all select-list entries resolve to disjoint properties exposed by one CIM class named in the from-clause.
1130 This rule does not prevent such queries from using joins.

1131 Neither the query language nor the format of the query is defined by this specification. It is anticipated
1132 that query languages will be submitted to the DMTF as separate proposals.

1133   CIM servers can declare which query languages they support (if any) using a mechanism defined in 7.5.

1134   If ExecQuery is successful, the method returns zero or more CIM classes or instances that correspond to
1135   the results of the query.

1136   If ExecQuery is unsuccessful, the method shall return one of the following status codes, where the error
1137   returned is the first applicable error in the list, starting with the first element and working down. Any
1138   additional method-specific interpretation of the error is enclosed in parentheses.

1139   • CIM_ERR_ACCESS_DENIED

1140   • CIM_ERR_NOT_SUPPORTED

1141   • CIM_ERR_INVALID_NAMESPACE

1142   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1143   incorrect parameters)

1144   • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested query language is not
1145   recognized.)

1146   • CIM_ERR_INVALID_QUERY (The query is not a valid query in the specified query language.)

1147   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1148   **5.3.2.14 Associators**

1149   The Associators operation enumerates CIM objects (classes or instances) associated with a particular
1150   source CIM object:

```
1151        Associators
1152        <objectWithPath>*Associators (
1153            [IN] <objectName> ObjectName,
1154            [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
1155            [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1156            [IN,OPTIONAL,NULL] string Role = NULL,
1157            [IN,OPTIONAL,NULL] string ResultRole = NULL,
1158            [IN,OPTIONAL] boolean IncludeQualifiers = false, (DEPRECATED)
1159            [IN,OPTIONAL] boolean IncludeClassOrigin = false,
1160            [IN,OPTIONAL,NULL] string PropertyList [] = NULL
1161        )
```

1162   The ObjectName input parameter defines the source CIM object whose associated objects are to be
1163   returned. This may be either a class name or instance name (model path).

1164   The AssocClass input parameter, if not NULL, shall be a valid CIM association class name. It acts as a
1165   filter on the returned set of objects by mandating that each returned object shall be associated to the
1166   source object through an instance of this class or one of its subclasses.

1167   The ResultClass input parameter, if not NULL, shall be a valid CIM class name. It acts as a filter on the
1168   returned set of objects by mandating that each returned object shall be either an instance of this class (or
1169   one of its subclasses) or be this class (or one of its subclasses).

1170   The Role input parameter, if not NULL, shall be a valid property name. It acts as a filter on the returned
1171   set of objects by mandating that each returned object shall be associated with the source object through
1172   an association in which the source object plays the specified role. That is, the name of the property in the
1173   association class that refers to the source object shall match the value of this parameter.

1174   The ResultRole input parameter, if not NULL, shall be a valid property name. It acts as a filter on the
1175   returned set of objects by mandating that each returned object shall be associated to the source object

1176 through an association in which the returned object plays the specified role. That is, the name of the
1177 property in the association class that refers to the returned object shall match the value of this parameter.

1178 **DEPRECATION NOTE:** The use of the `IncludeQualifiers` parameter is DEPRECATED and it may
1179 be removed in a future version of this specification. The preferred behavior is to use the class operations
1180 to receive qualifier information and not depend on any qualifiers in this response. If
1181 `IncludeQualifiers` is `true`, all qualifiers for each object (including qualifiers on the object and on any
1182 returned properties) shall be included as `<QUALIFIER>` elements in the response. If it is `false`, no
1183 `<QUALIFIER>` elements are present.

1184 If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on
1185 all appropriate elements in each returned object. If it is `false`, no `CLASSORIGIN` attributes are present.

1186 If the `PropertyList` input parameter is not NULL, the members of the array define one or more
1187 property names. Each returned object shall not include elements for any properties missing from this list.
1188 If `PropertyList` is an empty array, no properties are included in each returned object. If it is NULL, no
1189 additional filtering is defined.

1190 If `PropertyList` contains duplicate elements, the server shall ignore them but otherwise process the
1191 request normally. If `PropertyList` contains elements that are invalid property names for any target
1192 object, the server shall ignore such entries but otherwise process the request normally.

1193 Clients should not explicitly specify properties in the `PropertyList` parameter unless they specify a
1194 non-NULL value for the `ResultClass` parameter.

1195 If instances are returned, properties with the NULL value may be omitted from the response, even if the
1196 CIM client has not requested the exclusion of the through the `PropertyList` filter. The CIM client shall
1197 interpret such omitted properties as NULL. Note that the CIM client cannot make any assumptions about
1198 properties omitted as a result of using the `PropertyList` filter. If classes are returned, the CIM server
1199 cannot make this choice, and only the CIM client can cause properties to be excluded by using the
1200 `PropertyList` filter.

1201 If Associators is successful, the method returns zero or more CIM classes or instances meeting the
1202 requested criteria. Because it is possible for CIM objects from different hosts or namespaces to be
1203 associated, each returned object includes location information. If the ObjectName refers to a class, then
1204 classes are returned. These classes shall have all CIM elements (properties, methods, and qualifiers)
1205 defined in and inherited by that class, reduced by any properties excluded as a result of using the
1206 `PropertyList` filter. If the ObjectName refers to an instance, then instances are returned. These
1207 instances shall have all properties defined in and inherited by its class, reduced by any properties
1208 excluded as a result of using the `PropertyList` filter and further reduced by any NULL valued
1209 properties omitted from the response.

1210 If Associators is unsuccessful, this method shall return one of the following status codes, where the error
1211 returned is the first applicable error in the list, starting with the first element and working down. Any
1212 additional method-specific interpretation of the error is enclosed in parentheses.

1213 • CIM_ERR_ACCESS_DENIED

1214 • CIM_ERR_NOT_SUPPORTED (by the CIM server for this operation)

1215 • CIM_ERR_INVALID_NAMESPACE

1216 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1217   incorrect parameters)

1218 • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1219   instance, if provided.)

1220   • CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
1221   unspecified error occurred.)

1222   **5.3.2.15 AssociatorNames**

1223   The AssociatorNames operation enumerates the names of CIM Objects (classes or instances) that are
1224   associated with a particular source CIM object:

```
1225        AssociatorNames
1226        <objectPath>*AssociatorNames (
1227            [IN] <objectName> ObjectName,
1228            [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
1229            [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1230            [IN,OPTIONAL,NULL] string Role = NULL,
1231            [IN,OPTIONAL,NULL] string ResultRole = NULL
1232        )
```

1233   The ObjectName input parameter defines the source CIM object whose associated names are to be
1234   returned. This is either a class or instance name (model path).

1235   The AssocClass input parameter, if not NULL, shall be a valid CIM association class name. It acts as a
1236   filter on the returned set of names by mandating that each returned name identify an object that shall be
1237   associated to the source object through an instance of this class or one of its subclasses.

1238   The ResultClass input parameter, if not NULL, shall be a valid CIM class name. It acts as a filter on the
1239   returned set of names by mandating that each returned name identify an object that shall be either an
1240   instance of this class (or one of its subclasses) or be this class (or one of its subclasses).

1241   The Role input parameter, if not NULL, shall be a valid property name. It acts as a filter on the returned
1242   set of names by mandating that each returned name identify an object that shall be associated to the
1243   source object through an association in which the source object plays the specified role. That is, the
1244   name of the property in the association class that refers to the source object shall match the value of this
1245   parameter.

1246   The ResultRole input parameter, if not NULL, shall be a valid property name. It acts as a filter on the
1247   returned set of names by mandating that each returned name identify an object that shall be associated
1248   to the source object through an association in which the named returned object plays the specified role.
1249   That is, the name of the property in the association class that refers to the returned object shall match the
1250   value of this parameter.

1251   If AssociatorNames is successful, the method returns zero or more CIM class paths or instance paths
1252   meeting the requested criteria. Because CIM objects from different hosts or namespaces can be
1253   associated, each returned object includes location information. If the ObjectName refers to a class path,
1254   then class paths are returned. Otherwise, the ObjectName refers to an instance path, and instance paths
1255   are returned.

1256   If AssociatorNames is unsuccessful, one of the following status codes shall be returned by this method,
1257   where the first applicable error in the list (starting with the first element of the list, and working down) is
1258   the error returned. Any additional method-specific interpretation of the error is given in parentheses.

1259   • CIM_ERR_ACCESS_DENIED

1260   • CIM_ERR_NOT_SUPPORTED (by the CIM server for this operation)

1261   • CIM_ERR_INVALID_NAMESPACE

1262   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise
1263   incorrect parameters)

1264   • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1265   instance, if provided.)

1266   • CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
1267   unspecified error occurred.)

5.3.2.16 References

1269   The References operation enumerates the association objects that refer to a particular target CIM object
1270   (class or instance).

```
1271        References
1272        <objectWithPath>*References (
1273            [IN] <objectName> ObjectName,
1274            [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1275            [IN,OPTIONAL,NULL] string Role = NULL,
1276            [IN,OPTIONAL] boolean IncludeQualifiers = false,  (DEPRECATED)
1277            [IN,OPTIONAL] boolean IncludeClassOrigin = false,
1278            [IN,OPTIONAL,NULL] string PropertyList [] = NULL
1279        )
```

1280   The `ObjectName` input parameter defines the target CIM object whose referring objects are to be
1281   returned. This is either a class or instance name (model path).

1282   The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the
1283   returned set of objects by mandating that each returned object shall be an instance of this class (or one of
1284   its subclasses) or this class (or one of its subclasses).

1285   The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned
1286   set of objects by mandating that each returned object shall refer to the target object through a property
1287   with a name that matches the value of this parameter.

1288   **DEPRECATION NOTE:** The use of the `IncludeQualifiers` parameter is DEPRECATED and it may
1289   be removed in a future version of this specification. The preferred behavior is to use the class operations
1290   to receive qualifier information and not depend on any qualifiers in this response. If
1291   `IncludeQualifiers` is `true`, all qualifiers for each object (including qualifiers on the object and on any
1292   returned properties) shall be included as `<QUALIFIER>` elements in the response. If this parameter is
1293   `false`, no `<QUALIFIER>` elements are present in each returned Object.

1294   If the `IncludeClassOrigin` input parameter is `true`, the `CLASSORIGIN` attribute shall be present on
1295   all appropriate elements in each returned object. If it is `false`, no `CLASSORIGIN` attributes are present.

1296   If the `PropertyList` input parameter is not NULL, the members of the array define one or more
1297   property names. Each returned object shall not include elements for any properties missing from this list.
1298   If `PropertyList` is an empty array, no properties are included in each returned object. If
1299   `PropertyList` is NULL, no additional filtering is defined.

1300   If `PropertyList` contains duplicate elements, the server shall ignore them but otherwise process the
1301   request normally. If `PropertyList` contains elements that are invalid property names for any target
1302   object, the server shall ignore them but otherwise process the request normally.

1303   Clients should not explicitly specify properties in the `PropertyList` parameter unless they specify a
1304   non-NULL value for the `ResultClass` parameter.

1305   If instances are returned, properties with the NULL value may be omitted from the response, even if the
1306   CIM client has not requested the exclusion of the property through the `PropertyList` filter. The CIM
1307   client must interpret such omitted properties as NULL. Note that the CIM client cannot make any

1308 assumptions about properties omitted as a result of using the `PropertyList` filter. If classes are
1309 returned, the CIM server cannot make this choice, and only the CIM client can cause properties to be
1310 excluded by using the `PropertyList` filter.

1311 If References is successful, the method returns zero or more CIM classes or instances meeting the
1312 requested criteria. Because CIM objects from different hosts or namespaces can be associated, each
1313 returned object includes location information. If the ObjectName refers to a class, then classes are
1314 returned. These classes shall have all CIM elements (properties, methods, and qualifiers) defined in and
1315 inherited by that class, reduced by any properties excluded as a result of using the `PropertyList` filter.
1316 If the ObjectName refers to an instance, then instances are returned. These instances shall have all
1317 properties defined in and inherited by their respective classes, reduced by any properties excluded as a
1318 result of using the `PropertyList` filter and further reduced by any NULL valued properties omitted from
1319 the response.

1320 If References is unsuccessful, this method shall return one of the following status codes, where the error
1321 returned is the first applicable error in the list, starting with the first element and working down. Any
1322 additional method-specific interpretation of the error is enclosed in parentheses.

1323 • CIM_ERR_ACCESS_DENIED

1324 • CIM_ERR_NOT_SUPPORTED (by the CIM server for this operation)

1325 • CIM_ERR_INVALID_NAMESPACE

1326 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1327   incorrect parameters)

1328 • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1329   instance, if provided.)

1330 • CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
1331   unspecified error occurred.)

1332 5.3.2.17 ReferenceNames

1333 The ReferenceNames operation enumerates the association objects that refer to a particular target CIM
1334 object (class or instance):

```
1335         ReferenceNames
1336         <objectPath>*ReferenceNames (
1337             [IN] <objectName> ObjectName,
1338             [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1339             [IN,OPTIONAL,NULL] string Role = NULL
1340         )
```

1341 The `ObjectName` input parameter defines the target CIM object with the referring object names to be
1342 returned. It may be either a class or an instance name (model path).

1343 The `ResultClass` input parameter, if not `NULL`, shall be a valid CIM class name. It acts as a filter on the
1344 returned set of object names by mandating that each returned Object Name identify an instance of this
1345 class (or one of its subclasses) or this class (or one of its subclasses).

1346 The `Role` input parameter, if not `NULL`, shall be a valid property name. It acts as a filter on the returned
1347 set of object names by mandating that each returned object name shall identify an object that refers to the
1348 target instance through a property with a name that matches the value of this parameter.

1349 If ReferenceNames is successful, the method returns zero or more CIM class paths or instance paths
1350 meeting the requested criteria. Because CIM objects from different hosts or namespaces can be
1351 associated, each returned object includes location information. If the ObjectName refers to a class path,

1352   then class paths are returned. Otherwise, the ObjectName refers to an instance path, and instance paths
1353   are returned.

1354   If ReferenceNames is unsuccessful, this method shall return one of the following status codes, where the
1355   error returned is the first applicable error in the list, starting with the first element and working down. Any
1356   additional method-specific interpretation of the error is enclosed in parentheses.

1357   • CIM_ERR_ACCESS_DENIED

1358   • CIM_ERR_NOT_SUPPORTED (by the CIM server for this operation)

1359   • CIM_ERR_INVALID_NAMESPACE

1360   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1361   incorrect parameters)

1362   • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1363   instance, if provided.)

1364   • CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
1365   unspecified error occurred.)

1366   5.3.2.18 GetProperty

1367   The GetProperty operation retrieves a single property value from a CIM instance in the target
1368   namespace:

1369   ```
       GetProperty
1370   <propertyValue>GetProperty (
1371   [IN] <instanceName> InstanceName,
1372   [IN] string PropertyName
1373   )
       ```

1374   The `InstanceName` input parameter specifies the name of the instance (model path) from which the
1375   property value is requested.

1376   The `PropertyName` input parameter specifies the name of the property with the value to be returned.

1377   If GetProperty is successful, the return value specifies the value of the requested property. If the value is
1378   NULL, no element is returned.

1379   If GetProperty is unsuccessful, this method shall return one of the following status codes, where the error
1380   returned is the first applicable error in the list, starting with the first element and working down. Any
1381   additional method-specific interpretation of the error is enclosed in parentheses.

1382   • CIM_ERR_ACCESS_DENIED

1383   • CIM_ERR_INVALID_NAMESPACE

1384   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1385      incorrect parameters)

1386   • CIM_ERR_INVALID_CLASS (The CIM class does not exist in the specified namespace.)

1387   • CIM_ERR_NOT_FOUND (The CIM class exists, but the requested CIM instance does not exist
1388      in the specified namespace.)

1389   • CIM_ERR_NO_SUCH_PROPERTY (The CIM instance exists, but the requested property does
1390      not.)

1391   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1392    **5.3.2.19  SetProperty**

1393    The SetProperty operation sets a single property value in a CIM instance in the target namespace:

```
1394        SetProperty
1395        void SetProperty (
1396            [IN] <instanceName> InstanceName,
1397            [IN] string PropertyName,
1398            [IN,OPTIONAL,NULL] <propertyValue> NewValue = NULL
```

1399   The `InstanceName` input parameter specifies the name of the instance (model path) with the property
1400   value to be updated.

1401   The `PropertyName` input parameter specifies the name of the property with the value to be updated.

1402   The `NewValue` input parameter specifies the new value for the property (which may be NULL).

1403   If SetProperty is unsuccessful, this method shall return one of the following status codes, where the error
1404   returned is the first applicable error in the list, starting with the first element and working down. Any
1405   additional method-specific interpretation of the error is enclosed in parentheses.

1406   • CIM_ERR_ACCESS_DENIED

1407   • CIM_ERR_NOT_SUPPORTED (by the CIM server for this operation)

1408   • CIM_ERR_INVALID_NAMESPACE

1409   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1410   incorrect parameters)

1411   • CIM_ERR_INVALID_CLASS (The CIM class does not exist in the specified namespace.)

1412   • CIM_ERR_NOT_FOUND (The CIM class exists, but the requested CIM instance does not exist
1413   in the specified namespace.)

1414   • CIM_ERR_NOT_SUPPORTED (This operation is not supported for the class of the specified
1415   instance, if provided.)

1416   • CIM_ERR_NO_SUCH_PROPERTY (The CIM instance exists, but the requested property does
1417   not.)

1418   • CIM_ERR_TYPE_MISMATCH (The supplied value is incompatible with the type of the
1419   property.)

1420   • CIM_ERR_FAILED (This operation is not supported for the specified instance, or some other
1421   unspecified error occurred.)

1422   **5.3.2.20 GetQualifier**

1423   The GetQualifier operation retrieves a single qualifier declaration from the target namespace.

```
1424        GetQualifier
1425        <qualifierDecl>GetQualifier (
1426            [IN] string QualifierName
1427        )
```

1428   The `QualifierName` input parameter identifies the qualifier with the declaration to be retrieved.

1429   If GetQualifier is successful, the method returns the qualifier declaration for the named qualifier.

1430 If GetQualifier is unsuccessful, this method shall return one of the following status codes, where the error
1431 returned is the first applicable error in the list, starting with the first element and working down. Any
1432 additional method-specific interpretation of the error is enclosed in parentheses.

1433 • CIM_ERR_ACCESS_DENIED

1434 • CIM_ERR_NOT_SUPPORTED

1435 • CIM_ERR_INVALID_NAMESPACE

1436 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1437 incorrect parameters)

1438 • CIM_ERR_NOT_FOUND (The requested qualifier declaration does not exist.)

1439 • CIM_ERR_FAILED (Some other unspecified error occurred.)

1440 5.3.2.21 SetQualifier

1441 The SetQualifier operation creates or updates a single qualifier declaration in the target namespace. If the
1442 qualifier declaration already exists, it is overwritten:

```
1443    SetQualifier
1444    void SetQualifier (
1445        [IN] <qualifierDecl> QualifierDeclaration
1446    )
```

1447 The `QualifierDeclaration` input parameter defines the qualifier declaration to add to the
1448 namespace.

1449 If SetQualifier is successful, the qualifier declaration is added to the target namespace. If a qualifier
1450 declaration with the same qualifier name already exists, the new declaration replaces it.

1451 If SetQualifier is unsuccessful, this method returns one of the following status codes, where the error
1452 returned is the first applicable error in the list, starting with the first element and working down. Any
1453 additional method-specific interpretation of the error is enclosed in parentheses.

1454 • CIM_ERR_ACCESS_DENIED

1455 • CIM_ERR_NOT_SUPPORTED

1456 • CIM_ERR_INVALID_NAMESPACE

1457 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1458 incorrect parameters)

1459 • CIM_ERR_FAILED (Some other unspecified error occurred.)

1460 5.3.2.22 DeleteQualifier

1461 The DeleteQualifier operation deletes a single qualifier declaration from the target namespace.

```
1462    DeleteQualifier
1463    void DeleteQualifier (
1464        [IN] string QualifierName
1465    )
```

1466 The `QualifierName` input parameter identifies the qualifier with the declaration to be deleted.

1467 If DeleteQualifier is successful, the specified qualifier declaration is deleted from the namespace.

1468   If DeleteQualifier is unsuccessful, this method shall return one of the following status codes, where the
1469   error returned is the first applicable error in the list, starting with the first element and working down. Any
1470   additional method-specific interpretation of the error is enclosed in parentheses.

1471   • CIM_ERR_ACCESS_DENIED

1472   • CIM_ERR_NOT_SUPPORTED

1473   • CIM_ERR_INVALID_NAMESPACE

1474   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1475   incorrect parameters)

1476   • CIM_ERR_NOT_FOUND (The requested qualifier declaration does not exist.)

1477   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1478   **5.3.2.23 EnumerateQualifiers**

1479   The EnumerateQualifiers operation enumerates qualifier declarations from the target namespace.

1480   **EnumerateQualifiers**
1481   <qualifierDecl>***EnumerateQualifiers** (
1482   )

1483   If EnumerateQualifiers is successful, the method returns zero or more qualifier declarations.

1484   If EnumerateQualifiers is unsuccessful, this method shall return one of the following status codes, where
1485   the error returned is the first applicable error in the list, starting with the first element and working down.
1486   Any additional method-specific interpretation of the error is enclosed in parentheses.

1487   • CIM_ERR_ACCESS_DENIED

1488   • CIM_ERR_NOT_SUPPORTED

1489   • CIM_ERR_INVALID_NAMESPACE

1490   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1491   incorrect parameters)

1492   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1493   **5.3.2.24 Pulled Enumeration Operations**

1494   This clause defines a set of operations that return CIM instances or instance paths in portions controlled
1495   by the CIM client. These operations are called *pulled enumerations*. Usually, an enumeration session is
1496   established through an Open operation, and subsequent repeated executions of a Pull operation on the
1497   enumeration session are used to retrieve them. Optionally, the Open operation can also pull a first set of
1498   items.

1499   Pulled enumeration operations consist of the following individual operations:

1500   • Open operations open an enumeration of the following elements:

1501   – OpenEnumerateInstances (instances of a class)

1502   – OpenEnumerateInstancePaths (instance paths of instances of a class)

1503   – OpenReferenceInstances (association instances referencing a target instance)

1504   – OpenReferenceInstancePaths (the instance paths of association instances referencing a
1505   target instance)

1506   – OpenAssociatorInstances (instances associated with a source instance)

1507   –   OpenAssociatorInstancePaths (the instance paths of instances associated to a source
1508   instance)

1509   –   OpenQueryInstances (the rows resulting from a query)

1510   •   Pull operations are for the following cases:

1511   –   PullInstances (Instances are enumerated, and instance paths are either not available, for
1512   example as in for OpenQueryInstances, or not desired.)

1513   –   PullInstancesWithPath (Instances with paths are enumerated.)

1514   –   PullInstancePaths (Instance paths are enumerated.)

1515   •   Other operations are as follows:

1516   –   CloseEnumeration (closes an open enumeration)

1517   –   EnumerationCount (estimates the number of items in an open enumeration)

1518   **5.3.2.24.1 Behavioral Rules for Pulled Enumeration Operations**

1519   A central concept of pulled enumeration operations is the "enumeration session," which provides a
1520   context in which the operations perform their work and which determines the set of elements to be
1521   returned. To process the operations of an enumeration session, some parameters of the Open operation
1522   need to be maintained as long as the enumeration session is open. In addition, some state data about
1523   where the enumeration session is with regard to elements already returned must be maintained.

1524   From a CIM client perspective, an enumeration session is an enumeration context value. A successful
1525   Open operation establishes the enumeration session and returns an enumeration context value
1526   representing it. This value is used as an input/output parameter in subsequent Pull operations on that
1527   enumeration session. The enumeration context value shall uniquely identify the open enumeration
1528   session within the target CIM namespace of the Open operation that established the enumeration
1529   session. It is valid for a CIM server to use NULL as an enumeration context value representing a closed
1530   enumeration session, but a CIM client shall not rely on that.

1531   Defining the enumeration context value in Pull operations as both an input parameter and an output
1532   parameter allows the CIM server to change the enumeration context value during the execution of a pull
1533   operation. This ability to change allows different implementation approaches on the CIM server side,
1534   which are transparent for the CIM client. Example approaches are as follows:

1535   •   Maintain any state data describing the enumeration session internally in the CIM server. The
1536   enumeration context value does not need to change in subsequent Pull operations. The CIM
1537   server uses this value only to identify the internal state data for the open enumeration session. It
1538   does not use the value to store any state data. A variation of this approach is to hand back
1539   modified enumeration context values for additional CIM server-side sequence checking.

1540   •   Maintain any state data describing the enumeration session only on the CIM client side. All state
1541   data is stored in the enumeration context value, and the CIM server does not maintain any state
1542   data about the enumeration session, essentially being completely stateless with regard to the
1543   enumeration session.

1544   •   A combination of the two previous approaches.

1545   A CIM server may support keeping enumeration sessions open across connection terminations and
1546   shutdowns of the server. Elements may be created, deleted, or modified concurrently with an
1547   enumeration session that involves these elements. Such changes may or may not be reflected in the
1548   enumeration set. Therefore, there is no guarantee to the CIM client that the enumeration set represents a
1549   consistent snapshot of its elements at a point in time. However, the CIM server should make a best effort
1550   attempt for the returned enumeration set to represent a consistent snapshot of its elements at a point in
1551   time. The order of elements in the enumeration set is undefined.

1552 This specification does not restrict the number of enumeration sessions that can be established or
1553 executed concurrently in the same CIM server or client. This remains true even if the enumeration sets of
1554 such concurrently established enumeration sessions contain the same elements.

1555 Except for CloseEnumeration, all operations on a particular enumeration session shall be executed
1556 sequentially. An enumeration session can be open or closed. It is considered open if operations using its
1557 enumeration context value as an input parameter can be executed successfully. It is opened by the
1558 successful completion of an Open operation and closed by one of the following events:

1559 - Successful completion of a CloseEnumeration operation

1560 - Successful completion of an open or pull operation with the EndOfSequence output parameter
1561   set to `true`

1562 - Unsuccessful completion of a pull operation when ContinueOnError is not requested

1563 - CIM server-side decision to close the enumeration session based upon an operation timeout

1564 - CIM server-side decision to close an enumeration session during an operation on that
1565   enumeration session based upon exceeding server limits

1566 A conformant CIM server may support closure of enumeration sessions based upon exceeding server
1567 limits. Example situations for such a decision are:

1568 - Pull operations with no objects requested that are repeated with a high frequency on the same
1569   enumeration session

1570 - EnumerationCount operations repeated with a high frequency on the same enumeration
1571   session

1572 A mechanism by which CIM servers can declare support for closure of enumeration sessions based upon
1573 exceeding server limits is defined in 7.5. If a CIM server supports such closure of enumeration sessions, it
1574 shall make the decision to close during an operation on that enumeration session. There is no way to
1575 indicate the reason for the closure if the decision is made elsewhere. If a CIM server closes an
1576 enumeration session based upon exceeding server limits, it shall return failure on the operation on that
1577 enumeration session with the status code CIM_ERR_SERVER_LIMITS_EXCEEDED.

1578 **5.3.2.24.2 Common Parameters for the Open Operations**

1579 This clause defines commonly used parameters for the Open operations. The description of the individual
1580 Open operations references these parameters as appropriate. Note that not every Open operation uses
1581 every one of these common parameters:

1582 - EnumerationContext

1583   - This output parameter is the enumeration context value representing the enumeration
1584   session. If the EndOfSequence is true, the EnumerationContext value may be NULL.

1585 - EndOfSequence

1586   - This output parameter indicates to the CIM client whether the enumeration session is
1587   exhausted. If EndOfSequence is `true` upon successful completion of an Open operation,
1588   no more elements are available and the CIM server closes the enumeration session,
1589   releasing any allocated resources related to the enumeration session. If the enumeration
1590   set is empty, it is valid for a CIM server to set EndOfSequence to `true`, even if
1591   MaxObjectCount is 0. In this case, the enumeration session is closed upon successful
1592   completion of the Open operation. If EndOfSequence is `false`, additional elements may
1593   be available and the CIM server shall not close the enumeration session.

- IncludeClassOrigin

  – This input parameter is used only on Open operations that enumerate CIM instances. It controls whether information about the class origin of properties, references or methods is included in any enumerated CIM instances. If IncludeClassOrigin is `true`, the CLASSORIGIN attribute shall be present on all appropriate elements in each CIM instance returned by any subsequent PullInstance operations on this enumeration session. If IncludeClassOrigin is `false`, any CLASSORIGIN attributes shall not be present in any enumerated instances.

- FilterQueryLanguage and FilterQuery

  – These input parameters act as an additional restricting filter on the set of enumerated elements (for example, instances or instance paths).

  – If FilterQueryLanguage is not NULL, it shall specify a query language and FilterQuery shall be a valid query in that query language. Neither the query language nor the format of the query is defined by this specification. It is anticipated that query languages will be submitted to the DMTF as separate proposals. A mechanism by which CIM servers can declare the query languages they support for filtering in Pulled enumerations (if any) is defined in 7.5.

  – The query specified in FilterQuery shall address (for example, in its FROM list) the class specified in ClassName and shall not address any other classes. The result set specified by the query (for example, SELECT list) shall be ignored. The query shall not define any ordering criteria or any grouping of objects.

  – If the CIM server does not support filtered enumerations and FilterQueryLanguage is not NULL, the CIM server shall return a failure with the status code CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED.

- OperationTimeout

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

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

  – If OperationTimeout is NULL, the CIM server shall choose an operation timeout.

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

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

- ContinueOnError

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

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

1641	–	If a CIM server supports continuation on error and ContinueOnError is `true`, the
1642		enumeration session shall remain open when a Pull operation fails, and any subsequent
1643		successful Pull operations shall return the set of elements that would have been returned if
1644		the failing Pull operations were successful. This behavior is subject to the consistency rules
1645		defined for pulled enumerations. If ContinueOnError is `false`, the enumeration session
1646		shall be closed when a Pull operation returns a failure.

1647 • MaxObjectCount

1648	–	This input parameter defines the maximum number of elements that this Open operation
1649		can return. Any uint32 number is valid, including 0. The CIM server may deliver any
1650		number of elements up to MaxObjectCount but shall not deliver more
1651		than MaxObjectCount elements. A conformant CIM server implementation may choose to
1652		never return any elements during an Open operation, regardless of the value of
1653		MaxObjectCount. Note that a CIM client can use a MaxObjectCount value of 0 to specify
1654		that it does not want to retrieve any instances in the Open operation.

1655 • Return Value (array of enumerated elements)

1656	–	The return value of a successful Open operation is an array of enumerated elements with a
1657		number of entries from 0 up to a maximum defined by MaxObjectCount. These entries
1658		meet the criteria defined in the Open operation. Note that returning no entries in the array
1659		does not imply that the enumeration session is exhausted. Only the EndOfSequence
1660		output parameter indicates whether the enumeration session is exhausted.

1661 **5.3.2.24.3 OpenEnumerateInstances**

1662 The OpenEnumerateInstances operation establishes and opens an enumeration session of the instances
1663 of a CIM class (including instances of its subclasses) in the target namespace. Optionally, it retrieves a
1664 first set of instances.

```
1665          OpenEnumerateInstances
1666          <instanceWithPath>* OpenEnumerateInstances (
1667              [OUT] <enumerationContext> EnumerationContext,
1668              [OUT] Boolean EndOfSequence,
1669              [IN] <className> ClassName,
1670              [IN,OPTIONAL] boolean DeepInheritance = true,
1671              [IN,OPTIONAL] boolean IncludeClassOrigin = false,
1672              [IN,OPTIONAL,NULL] string PropertyList [] = NULL,
1673              [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
1674              [IN,OPTIONAL,NULL] string FilterQuery = NULL,
1675              [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
1676              [IN,OPTIONAL] Boolean ContinueOnError = false,
1677              [IN,OPTIONAL] uint32 MaxObjectCount = 0
1678          )
```

1679 The OpenEnumerateInstances operation shall comply with the behavior defined in 5.3.2.24.1.

1680 The `EnumerationContext` output parameter is defined in 5.3.2.24.2.

1681 The `EndOfSequence` output parameter is defined in 5.3.2.24.2.

1682 The `ClassName` input parameter defines the class that is the basis for the enumeration. The enumeration
1683 set shall consist of all instances of that specified class, including any instances of any of its subclasses, in
1684 accordance with the polymorphic nature of CIM objects.

1685 The `DeepInheritance` input parameter acts as a filter on the properties included in any enumerated
1686 CIM instances. If the `DeepInheritance` input parameter is `true`, all properties of each enumerated

1687 instance of the class shall be present (subject to constraints imposed by the other parameters), including
1688 any added by subclassing the specified class. If `DeepInheritance` is `false`, each enumerated
1689 instance includes only properties defined for the class specified by `ClassName`.

1690 The `IncludeClassOrigin` input parameter is defined in 5.3.2.24.2.

1691 The `PropertyList` input parameter acts as a filter on the properties in any enumerated CIM
1692 instances. If `PropertyList` is not NULL, the members of the array define zero or more property names
1693 of the specified class. This array may include inherited property names or property names explicitly
1694 defined in the specified class. However, it shall not include property names defined in subclasses of the
1695 specified class. Each enumerated instance shall not include elements for properties missing from this list.
1696 Note that `PropertyList` acts as an additional filter on the properties defined by the `DeepInheritance`
1697 input parameter. If `PropertyList` includes a property that is not in the set defined by
1698 `DeepInheritance`, the element for the property shall not be included. If `PropertyList` is an empty
1699 array, no elements for properties are included in the enumerated instances. If `PropertyList` is NULL,
1700 no additional filtering is defined. If `PropertyList` contains duplicate elements, the CIM server shall
1701 ignore them but otherwise process the request normally. If `PropertyList` contains elements that are
1702 invalid property names for any target instance, the CIM server shall ignore such entries but otherwise
1703 process the request normally.

1704 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.3.2.24.2.

1705 The `OperationTimeout` input parameter is defined in 5.3.2.24.2.

1706 The `ContinueOnError` input parameter is defined in 5.3.2.24.2.

1707 The `MaxObjectCount` input parameter is defined in 5.3.2.24.2.

1708 If OpenEnumerateInstances is successful, the return value shall be an array of enumerated instances as
1709 defined in 5.3.2.24.2.

1710 The PullInstancesWithPath operation shall be used to pull instances for an enumeration session opened
1711 using OpenEnumerateInstances. If any other operation is used to pull instances, the CIM server shall
1712 return failure with the status code CIM_ERR_FAILED.

1713 If OpenEnumerateInstances is unsuccessful, this operation shall return one of the following status codes,
1714 where the error returned is the first applicable error in the list, starting with the first element and working
1715 down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

1716 • CIM_ERR_ACCESS_DENIED

1717 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

1718 • CIM_ERR_NOT_SUPPORTED

1719 • CIM_ERR_INVALID_NAMESPACE

1720 • CIM_ERR_INVALID_OPERATION_TIMEOUT

1721 • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

1722 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1723   incorrect parameters)

1724 • CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not
1725   exist.)

1726 • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

1727 • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
1728   not recognized.)

1729   • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
1730   language.)

1731   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1732   5.3.2.24.4 OpenEnumerateInstancePaths

1733   The OpenEnumerateInstancePaths operation establishes and opens an enumeration session of the
1734   instance paths of the instances of a CIM class (including instances of its subclasses) in the target
1735   namespace. Optionally, it retrieves a first set of instance paths:

```
1736        OpenEnumerateInstancePaths
1737        <instancePath>* OpenEnumerateInstancePaths (
1738            [OUT] <enumerationContext> EnumerationContext,
1739            [OUT] Boolean EndOfSequence,
1740            [IN] <className> ClassName,
1741            [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
1742            [IN,OPTIONAL,NULL] string FilterQuery = NULL,
1743            [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
1744            [IN,OPTIONAL] Boolean ContinueOnError = false,
1745            [IN,OPTIONAL] uint32 MaxObjectCount = 0
1746        )
```

1747   The OpenEnumerateInstancePaths operation shall comply with the behavior defined in 5.3.2.24.1.

1748   The `EnumerationContext` output parameter is defined in 5.3.2.24.2.

1749   The `EndOfSequence` output parameter is defined in 5.3.2.24.2.

1750   The `ClassName` input parameter defines the class that is the basis for the enumeration. The
1751   enumeration set shall consist of the instance paths of all instances of the specified class, including any
1752   instances of any of its subclasses, in accordance with the polymorphic nature of CIM objects.

1753   The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.3.2.24.2.

1754   The `OperationTimeout` input parameter is defined in 5.3.2.24.2.

1755   The `ContinueOnError` input parameter is defined in 5.3.2.24.2.

1756   The `MaxObjectCount` input parameter is defined in 5.3.2.24.2.

1757   If OpenEnumerateInstancePaths is successful, the return value shall be an array of enumerated instance
1758   paths as defined in 5.3.2.24.2.

1759   The PullInstancePaths operation shall be used to pull instances for an enumeration session opened using
1760   OpenEnumerateInstancePaths. If any other operation is used to pull instances, the CIM server shall
1761   return failure with the status code CIM_ERR_FAILED.

1762   If OpenEnumerateInstancePaths is unsuccessful, this operation shall return one of the following status
1763   codes, where the error returned is the first applicable error in the list, starting with the first element and
1764   working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

1765   • CIM_ERR_ACCESS_DENIED

1766   • CIM_ERR_SERVER_IS_SHUTTING_DOWN

1767   • CIM_ERR_NOT_SUPPORTED

1768   • CIM_ERR_INVALID_NAMESPACE

1769 • CIM_ERR_INVALID_OPERATION_TIMEOUT

1770 • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

1771 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1772 incorrect parameters)

1773 • CIM_ERR_INVALID_CLASS (The CIM class that is the basis for this enumeration does not
1774 exist.)

1775 • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

1776 • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
1777 not recognized.)

1778 • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
1779 language.)

1780 • CIM_ERR_FAILED (Some other unspecified error occurred.)

1781 **5.3.2.24.5 OpenReferenceInstances**

1782 The OpenReferenceInstances operation establishes and opens the enumeration session of association
1783 instances that refer to a particular target CIM instance in the target namespace. Optionally, it retrieves a
1784 first set of instances:

```
1785        OpenReferenceInstances
1786        <instanceWithPath>* OpenReferenceInstances (
1787            [OUT] <enumerationContext> EnumerationContext,
1788            [OUT] Boolean EndOfSequence,
1789            [IN] <instanceName> InstanceName,
1790            [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1791            [IN,OPTIONAL,NULL] string Role = NULL,
1792            [IN,OPTIONAL] boolean IncludeClassOrigin = false,
1793            [IN,OPTIONAL,NULL] string PropertyList [] = NULL,
1794            [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
1795            [IN,OPTIONAL,NULL] string FilterQuery = NULL,
1796            [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
1797            [IN,OPTIONAL] Boolean ContinueOnError = false,
1798            [IN,OPTIONAL] uint32 MaxObjectCount = 0
1799        )
```

1800 The OpenReferenceInstances operation shall comply with the behavior defined in 5.3.2.24.1.

1801 The `EnumerationContext` output parameter is defined in 5.3.2.24.2.

1802 The `EndOfSequence` output parameter is defined in 5.3.2.24.2.

1803 The `InstanceName` input parameter specifies an instance name (model path) that identifies the target
1804 CIM instance with the referring association instances to be enumerated. Unless restricted by any of the
1805 filter parameters of this operation, the enumeration set shall consist of all association instances that
1806 reference the target instance.

1807 The `ResultClass` input parameter, if not NULL, shall be a CIM class name. It acts as a filter on the
1808 enumerated set of instances by mandating that each enumerated instance shall be an instance of this
1809 class or one of its subclasses. The CIM server shall not return an error if the `ResultClass` input
1810 parameter value is an invalid class name or if the class does not exist in the target namespace,

1811 The `Role` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the enumerated set
1812 of instances by mandating that each enumerated instance shall refer to the target instance through a

1813   property with a name that matches the value of this parameter. The CIM server shall not return an error if
1814   the `Role` input parameter value is an invalid property name or if the property does not exist,

1815   The `IncludeClassOrigin` input parameter is defined in 5.3.2.24.2.

1816   The `PropertyList` input parameter acts as a filter on the properties included in any enumerated CIM
1817   instances. If `PropertyList` is not `NULL`, the members of the array define zero or more property names.
1818   Each enumerated instance shall not include elements for any properties missing from this list.
1819   If `PropertyList` is an empty array, no properties are included in each enumerated instance. If
1820   `PropertyList` is `NULL`, all properties are included in each enumerated instance, subject to the
1821   conditions expressed by the other parameters. If `PropertyList` contains duplicate elements, the CIM
1822   server shall ignore them but otherwise process the request normally. If `PropertyList` contains
1823   elements that are invalid property names for any target instance, the CIM server shall ignore them but
1824   otherwise process the request normally. CIM clients should not specify properties in `PropertyList`
1825   unless they specify a non-NULL value for the `ResultClass` parameter.

1826   The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.3.2.24.2.

1827   The `OperationTimeout` input parameter is defined in 5.3.2.24.2.

1828   The `ContinueOnError` input parameter is defined in 5.3.2.24.2.

1829   The `MaxObjectCount` input parameter is defined in 5.3.2.24.2.

1830   If OpenReferenceInstances is successful, the return value shall be an array of enumerated instances as
1831   defined in 5.3.2.24.2.

1832   The PullInstancesWithPath operation shall be used to pull instances for an enumeration session opened
1833   using OpenReferenceInstances. If any other operation is used to pull instances, the CIM server shall
1834   return failure with the status code CIM_ERR_FAILED.

1835   If OpenReferenceInstances is unsuccessful, this operation shall return one of the following status codes,
1836   where the error returned is the first applicable error in the list, starting with the first element of and working
1837   down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

1838   • CIM_ERR_ACCESS_DENIED

1839   • CIM_ERR_SERVER_IS_SHUTTING_DOWN

1840   • CIM_ERR_NOT_SUPPORTED

1841   • CIM_ERR_INVALID_NAMESPACE

1842   • CIM_ERR_INVALID_OPERATION_TIMEOUT

1843   • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

1844   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized or otherwise
1845   incorrect parameters)

1846   • CIM_ERR_NOT_FOUND (The target instance was not found.)

1847   • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

1848   • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
1849   not recognized.)

1850   • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
1851   language.)

1852   • CIM_ERR_FAILED (Some other unspecified error occurred.)

1853 **5.3.2.24.6 OpenReferenceInstancePaths**

1854 The OpenReferenceInstancePaths operation establishes and opens an enumeration session of the
1855 instance paths of the association instances that refer to a particular target CIM instance in the target
1856 namespace. Optionally, it retrieves a first set of instance paths.

```
1857         OpenReferenceInstancePaths
1858         <instancePath>* OpenReferenceInstancePaths (
1859             [OUT] <enumerationContext> EnumerationContext,
1860             [OUT] Boolean EndOfSequence,
1861             [IN] <instanceName> InstanceName,
1862             [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1863             [IN,OPTIONAL,NULL] string Role = NULL,
1864             [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
1865             [IN,OPTIONAL,NULL] string FilterQuery = NULL,
1866             [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
1867             [IN,OPTIONAL] Boolean ContinueOnError = false,
1868             [IN,OPTIONAL] uint32 MaxObjectCount = 0
1869         )
```

1870 The OpenReferenceInstancePaths operation shall comply with the behavior defined in 5.3.2.24.1.

1871 The `EnumerationContext` output parameter is defined in 5.3.2.24.2.

1872 The `EndOfSequence` output parameter is defined in 5.3.2.24.2.

1873 The `InstanceName` input parameter specifies an instance name (model path) that identifies the target
1874 CIM instance with the referring association instances (respectively, their instance paths) to be
1875 enumerated. Unless restricted by any filter parameters of this operation, the enumeration set shall consist
1876 of the instance paths of all association instances that reference the target instance.

1877 The `ResultClass` input parameter, if not NULL, shall be a CIM class name. It acts as a filter on the
1878 enumerated set of instance paths by mandating that each enumerated instance path shall identify an
1879 instance of this class or one of its subclasses. The CIM server shall not return an error if the
1880 `ResultClass` input parameter value is an invalid class name or if the class does not exist in the target
1881 namespace.

1882 The `Role` input parameter, if not NULL, shall be a property name. It acts as a filter on the enumerated set
1883 of instance paths by mandating that each enumerated instance path shall identify an instance that refers
1884 to the target instance through a property with a name that matches the value of this parameter. The CIM
1885 server shall not return an error if the `Role` input parameter value is an invalid property name or if the
1886 property does not exist,

1887 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.3.2.24.2.

1888 The `OperationTimeout` input parameter is defined in 5.3.2.24.2.

1889 The `ContinueOnError` input parameter is defined in 5.3.2.24.2.

1890 The `MaxObjectCount` input parameter is defined in 5.3.2.24.2.

1891 If OpenReferenceInstancePaths is successful, the return value shall be an array of enumerated instance
1892 paths as defined in 5.3.2.24.2.

1893 The PullInstancePaths operation shall be used to pull instances for an enumeration session opened using
1894 OpenReferenceInstancePaths. If any other operation is used to pull instances, the CIM server shall return
1895 failure with the status code CIM_ERR_FAILED.

1896 If OpenReferenceInstancePaths is unsuccessful, this operation shall return one of the following status
1897 codes, where the error returned is the first applicable error in the list, starting with the first element and
1898 working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

1899 • CIM_ERR_ACCESS_DENIED

1900 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

1901 • CIM_ERR_NOT_SUPPORTED

1902 • CIM_ERR_INVALID_NAMESPACE

1903 • CIM_ERR_INVALID_OPERATION_TIMEOUT

1904 • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

1905 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1906   incorrect parameters)

1907 • CIM_ERR_NOT_FOUND (The target instance was not found.)

1908 • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

1909 • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
1910   not recognized.)

1911 • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
1912   language.)

1913 • CIM_ERR_FAILED (Some other unspecified error occurred.)

1914 **5.3.2.24.7 OpenAssociatorInstances**

1915 The OpenAssociatorInstances operation establishes and opens an enumeration session of the instances
1916 associated with a particular source CIM instance in the target namespace. Optionally, it retrieves a first
1917 set of instances.

```
1918        OpenAssociatorInstances
1919        <instanceWithPath>* OpenAssociatorInstances (
1920            [OUT] <enumerationContext> EnumerationContext,
1921            [OUT] Boolean EndOfSequence,
1922            [IN] <instanceName> InstanceName,
1923            [IN,OPTIONAL,NULL] <className> AssocClass = NULL,
1924            [IN,OPTIONAL,NULL] <className> ResultClass = NULL,
1925            [IN,OPTIONAL,NULL] string Role = NULL,
1926            [IN,OPTIONAL,NULL] string ResultRole = NULL,
1927            [IN,OPTIONAL] boolean IncludeClassOrigin = false,
1928            [IN,OPTIONAL,NULL] string PropertyList [] = NULL,
1929            [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
1930            [IN,OPTIONAL,NULL] string FilterQuery = NULL,
1931            [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
1932            [IN,OPTIONAL] Boolean ContinueOnError = false,
1933            [IN,OPTIONAL] uint32 MaxObjectCount = 0
1934        )
```

1935 The OpenAssociatorInstances operation shall comply with the behavior defined in 5.3.2.24.1.

1936 The `EnumerationContext` output parameter is defined in 5.3.2.24.2.

1937 The `EndOfSequence` output parameter is defined in 5.3.2.24.2.

1938 The `InstanceName` input parameter specifies an instance name (model path) that identifies the source
1939 CIM instance with the associated instances to be enumerated. Unless restricted by any filter parameters
1940 of this operation, the enumeration set shall consist of all instances associated with the source instance.

1941 The `AssocClass` input parameter, if not `NULL`, shall be a CIM association class name. It acts as a filter
1942 on the enumerated set of instances by mandating that each enumerated instance shall be associated with
1943 the source instance through an instance of this class or one of its subclasses. The CIM server shall not
1944 return an error if the `AssocClass` input parameter value is an invalid class name or if the class does not
1945 exist in the target namespace.

1946 The `ResultClass` input parameter, if not `NULL`, must be a CIM class name. It acts as a filter on the
1947 enumerated set of instances by mandating that each enumerated instance shall be an instance of this
1948 class or one of its subclasses. The CIM server shall not return an error if the `ResultClass` input
1949 parameter value is an invalid class name or if the class does not exist in the target namespace.

1950 The `Role` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the enumerated set
1951 of instances by mandating that each enumerated instance shall be associated with the source instance
1952 through an association in which the source instance plays the specified role. That is, the name of the
1953 property in the association class that refers to the source instance shall match the value of this
1954 parameter. The CIM server shall not return an error if the `Role` input parameter value is an invalid
1955 property name or if the property does not exist.

1956 The `ResultRole` input parameter, if not `NULL`, shall be a property name. It acts as a filter on the
1957 enumerated set of instances by mandating that each enumerated instance shall be associated with the
1958 source instance through an association in which the enumerated instance plays the specified role. That
1959 is, the name of the property in the association class that refers to the enumerated instance shall match
1960 the value of this parameter. The CIM server shall not return an error if the `ResultRole` input parameter
1961 value is an invalid property name or if the property does not exist.

1962 The `IncludeClassOrigin` input parameter is defined in 5.3.2.24.2.

1963 The `PropertyList` input parameter acts as a filter on the properties included in any enumerated CIM
1964 instances. If `PropertyList` is not NULL, the members of the array define zero or more property names.
1965 Each enumerated instance shall not include elements for any properties missing from this list.
1966 If `PropertyList` is an empty array, no properties are included in each enumerated instance. If
1967 `PropertyList` is NULL, all properties are included in each enumerated instance, subject to the
1968 conditions expressed by the other parameters. If `PropertyList` contains duplicate elements, the CIM
1969 server shall ignore them but otherwise process the request normally. If `PropertyList` contains
1970 elements that are invalid property names for any target instance, the CIM server shall ignore them but
1971 otherwise process the request normally. CIM clients should not specify properties in `PropertyList`
1972 unless they specify a non-NULL value for the ResultClass parameter.

1973 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.3.2.24.2.

1974 The `OperationTimeout` input parameter is defined in 5.3.2.24.2.

1975 The `ContinueOnError` input parameter is defined in 5.3.2.24.2.

1976 The `MaxObjectCount` input parameter is defined in 5.3.2.24.2.

1977 If OpenAssociatorInstances is successful, the return value shall be an array of enumerated instances as
1978 defined in 5.3.2.24.2.

1979 The PullInstancesWithPath operation shall be used to pull instances for an enumeration session opened
1980 using OpenAssociatorInstances. If any other operation is used to pull instances, the CIM server shall
1981 return failure with the status code CIM_ERR_FAILED.

1982 If OpenAssociatorInstances is unsuccessful, this operation shall return one of the following status codes,
1983 where the error returned is the first applicable error in the list, starting with the first element and working
1984 down. Any additional operation-specific interpretation of the error is given in parentheses.

1985 • CIM_ERR_ACCESS_DENIED

1986 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

1987 • CIM_ERR_NOT_SUPPORTED

1988 • CIM_ERR_INVALID_NAMESPACE

1989 • CIM_ERR_INVALID_OPERATION_TIMEOUT

1990 • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

1991 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
1992 incorrect parameters)

1993 • CIM_ERR_NOT_FOUND (The source instance was not found.)

1994 • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

1995 • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
1996 not recognized.)

1997 • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
1998 language.)

1999 • CIM_ERR_FAILED (Some other unspecified error occurred.)

2000 **5.3.2.24.8 OpenAssociatorInstancePaths**

2001 The OpenAssociatorInstancePaths operation establishes and opens an enumeration session of the
2002 instance paths of the instances associated with a particular source CIM instance in the target namespace.
2003 Optionally, it retrieves a first set of instance paths.

```
2004        OpenAssociatorInstancePaths
2005        <instancePath>* OpenAssociatorInstancePaths (
2006            [OUT] <enumerationContext>EnumerationContext,
2007            [OUT] Boolean EndOfSequence,
2008            [IN] <instanceName> InstanceName,
2009            [IN,OPTIONAL,NULL] <className>AssocClass= NULL,
2010            [IN,OPTIONAL,NULL] <className>ResultClass = NULL,
2011            [IN,OPTIONAL,NULL] string Role = NULL,
2012            [IN,OPTIONAL,NULL] string ResultRole = NULL,
2013            [IN,OPTIONAL,NULL] string FilterQueryLanguage = NULL,
2014            [IN,OPTIONAL,NULL] string FilterQuery = NULL,
2015            [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
2016            [IN,OPTIONAL] Boolean ContinueOnError = false,
2017            [IN,OPTIONAL] uint32 MaxObjectCount = 0
2018        )
```

2019 This operation shall comply with the behavior defined in 5.3.2.24.1.

2020 The `EnumerationContext` output parameter is defined in 5.3.2.24.2.

2021 The `EndOfSequence` output parameter is defined in 5.3.2.24.2.

2022 The `InstanceName` input parameter specifies an instance name (model path) that identifies the source
2023 CIM instance with the associated instances (respectively, their instance paths) to be enumerated. Unless

2024 restricted by any filter parameters of this operation, the enumeration set shall consist of the instance
2025 paths of all instances associated with the source instance.

2026 The `AssocClass` input parameter, if not `NULL`, shall be a CIM association class name. It acts as a filter
2027 on the enumerated set of instance paths by mandating that each instance path identify an instance that
2028 shall be associated with the source instance through an instance of this class or one of its subclasses.
2029 The CIM server shall not return an error if the `AssocClass` input parameter value is an invalid class
2030 name or if the class does not exist in the target namespace.

2031 The `ResultClass` input parameter, if not NULL, shall be a CIM class name. It acts as a filter on the
2032 enumerated set of instance paths by mandating that each instance path identify an instance that shall be
2033 an instance of this class or one of its subclasses. The CIM server shall not return an error if the
2034 `ResultClass` input parameter value is an invalid class name or if the class does not exist in the target
2035 namespace.

2036 The `Role` input parameter, if not NULL, shall be a property name. It acts as a filter on the enumerated set
2037 of instance paths by mandating that each instance path identify an instance that shall be associated with
2038 the source instance through an association in which the source instance plays the specified role. That is,
2039 the name of the property in the association class that refers to the source instance shall match the value
2040 of this parameter. The CIM server shall not return an error if the `Role` input parameter value is an invalid
2041 property name or if the property does not exist.

2042 The `ResultRole` input parameter, if not NULL, shall be a property name. It acts as a filter on the
2043 enumerated set of instance paths by mandating that each instance path identify an instance that shall be
2044 associated with the source instance through an association in which the instance identified by
2045 the enumerated instance path plays the specified role. That is, the name of the property in the association
2046 class that refers to the instance identified by the enumerated instance path shall match the value of this
2047 parameter. The CIM server shall not return an error if the `ResultRole` input parameter value is an
2048 invalid property name or if the property does not exist.

2049 The `FilterQueryLanguage` and `FilterQuery` input parameters are defined in 5.3.2.24.2.

2050 The `OperationTimeout` input parameter is defined in 5.3.2.24.2.

2051 The `ContinueOnError` input parameter is defined in 5.3.2.24.2.

2052 The `MaxObjectCount` input parameter is defined in 5.3.2.24.2.

2053 If OpenAssociatorInstancePaths is successful, the return value shall be an array of enumerated instance
2054 paths as defined in 5.3.2.24.2.

2055 The PullInstancePaths operation shall be used to pull instances for an enumeration session opened using
2056 OpenAssociatorInstancePaths. If any other operation is used to pull instances, the CIM server shall return
2057 failure with the status code CIM_ERR_FAILED.

2058 If OpenAssociatorInstancePaths is unsuccessful, this operation shall return one of the following status
2059 codes, where the error returned is the first applicable error in the list, starting with the first element and
2060 working down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

2061 • CIM_ERR_ACCESS_DENIED

2062 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2063 • CIM_ERR_NOT_SUPPORTED

2064 • CIM_ERR_INVALID_NAMESPACE

2065 • CIM_ERR_INVALID_OPERATION_TIMEOUT

2066 • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

2067   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2068   incorrect parameters)

2069   • CIM_ERR_NOT_FOUND (The source instance was not found.)

2070   • CIM_ERR_FILTERED_ENUMERATION_NOT_SUPPORTED

2071   • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
2072   not recognized.)

2073   • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter
2074   language.)

2075   • CIM_ERR_FAILED (Some other unspecified error occurred.)

2076   **5.3.2.24.9 Common Parameters for the Pull Operations**

2077   This clause defines commonly used parameters for the Pull operations. The description of the individual
2078   Pull operations references these parameters as appropriate. Note that not every Pull operation uses
2079   every one of these common parameters.

2080   • EnumerationContext

2081   – This parameter is the enumeration context value representing the enumeration session to
2082   be used.

2083   – When the Pull operation is invoked, the enumeration session represented by the
2084   `EnumerationContext` input parameter shall be open. The first enumeration session shall
2085   use one of the Open operations with a type of enumerated element that matches the Pull
2086   operation. For the first Pull operation on an enumeration session, the value of the
2087   `EnumerationContext` input parameter shall be the enumeration context value returned
2088   by a successful Open operation. For subsequent Pull operations on that enumeration
2089   session, the value of the `EnumerationContext` input parameter shall be the value of the
2090   `EnumerationContext` output parameter returned by the previous Pull operation on the
2091   same enumeration session.

2092   – After the Pull operation is completed, the enumeration session represented by the
2093   `EnumerationContext` output parameter shall be open or closed.

2094   • EndOfSequence

2095   – This output parameter indicates to the CIM client whether the enumeration session is
2096   exhausted. If `EndOfSequence` is `true` upon successful completion of a Pull operation, no
2097   more elements are available and the CIM server shall close the enumeration session,
2098   releasing any allocated resources related to the session. If `EndOfSequence` is `false`,
2099   additional elements may be available, and the CIM server shall not close the session.

2100   • MaxObjectCount

2101   – This input parameter defines the maximum number of elements that may be returned
2102   by this Pull operation. Any uint32 number is valid, including 0. The CIM server may deliver
2103   any number of elements up to `MaxObjectCount` but shall not deliver more than
2104   `MaxObjectCount` elements. The CIM client may use a MaxObjectCount value of 0 to
2105   restart the OperationTimeout for the enumeration session when it does not need to not
2106   retrieve any elements.

2107   • Return Value (array of enumerated elements)

2108   – The return value of a Pull operation upon successful completion is an array of enumerated
2109   elements with a number of entries from 0 up to a maximum defined by `MaxObjectCount`.
2110   These entries meet the criteria defined in the Open operation that established this
2111   enumeration session. Note that returning no entries in the array does not imply that the

2112   enumeration session is exhausted. Only the `EndOfSequence` output parameter indicates
2113   whether the enumeration session is exhausted.

2114 **5.3.2.24.10 PullInstancesWithPath**

2115 The PullInstancesWithPath operation retrieves instances including their instance paths from an open
2116 enumeration session represented by an enumeration context value:

```
2117      PullInstancesWithPath
2118      <instanceWithPath>* PullInstancesWithPath (
2119          [IN,OUT] <enumerationContext> EnumerationContext,
2120          [OUT] Boolean EndOfSequence,
2121          [IN] uint32 MaxObjectCount
2122      )/td>
```

2123 The PullInstancesWithPath operation shall comply with the behavior defined in 5.3.2.24.1.

2124 The `EnumerationContext` input/output parameter is defined in 5.3.2.24.9. The enumeration session
2125 shall be established using one of the OpenEnumerateInstances, OpenReferenceInstances, or
2126 OpenAssociatorInstances operations.

2127 The `EndOfSequence` output parameter is defined in 5.3.2.24.9.

2128 The `MaxObjectCount` input parameter is defined in 5.3.2.24.9.

2129 If PullInstancesWithPath is successful, the return value shall be an array of enumerated instances
2130 including their instance paths as defined in 5.3.2.24.9.

2131 If PullInstancesWithPath is unsuccessful, this operation shall return one of the following status codes,
2132 where the error returned is the first applicable error in the list, starting with the first element and working
2133 down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

2134 • CIM_ERR_ACCESS_DENIED

2135 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2136 • CIM_ERR_NOT_SUPPORTED

2137 • CIM_ERR_INVALID_NAMESPACE

2138 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2139 incorrect parameters)

2140 • CIM_ERR_INVALID_ENUMERATION_CONTEXT

2141 • CIM_ERR_PULL_HAS_BEEN_ABANDONED

2142 • CIM_ERR_SERVER_LIMITS_EXCEEDED

2143 • CIM_ERR_FAILED (Some other unspecified error occurred.)

2144 **5.3.2.24.11 PullInstancePaths**

2145 The PullInstancePaths operation retrieves instance paths from an open enumeration session represented
2146 by an enumeration context value:

```
2147      PullInstancePaths
2148      <instancePath>* PullInstancePaths (
2149          [IN,OUT] <enumerationContext> EnumerationContext,
2150          [OUT] Boolean EndOfSequence,
2151          [IN] uint32 MaxObjectCount
2152      )
```

2153   The PullInstancePaths operation shall comply with the behavior defined in 5.3.2.24.1.

2154   The `EnumerationContext` input/output parameter is defined in 5.3.2.24.9. The enumeration session
2155   shall have been established using one of the OpenEnumerateInstancePaths,
2156   OpenReferenceInstancePaths, or OpenAssociatorInstancePaths operations.

2157   The `EndOfSequence` output parameter is defined in 5.3.2.24.9.

2158   The `MaxObjectCount` input parameter is defined in 5.3.2.24.9.

2159   If PullInstancePaths is successful, the return value shall be an array of enumerated instance paths as
2160   defined in 5.3.2.24.9.

2161   If PullInstancePaths is unsuccessful, this operation shall return one of the following status codes, where
2162   the error returned is the first applicable error in the list, starting with the first element and working down.
2163   Any additional operation-specific interpretation of the error is enclosed in parentheses.

2164   • CIM_ERR_ACCESS_DENIED

2165   • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2166   • CIM_ERR_NOT_SUPPORTED

2167   • CIM_ERR_INVALID_NAMESPACE

2168   • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2169   incorrect parameters)

2170   • CIM_ERR_INVALID_ENUMERATION_CONTEXT

2171   • CIM_ERR_SERVER_LIMITS_EXCEEDED

2172   • CIM_ERR_PULL_HAS_BEEN_ABANDONED

2173   • CIM_ERR_FAILED (Some other unspecified error occurred.)

2174   **5.3.2.24.12 CloseEnumeration**

2175   The CloseEnumeration operation closes an open enumeration session, performing an early termination of
2176   an enumeration sequence:

2177   **CloseEnumeration**
2178   void **CloseEnumeration** (
2179   [IN] <u><enumerationContext></u> EnumerationContext
2180   )

2181   The `EnumerationContext` parameter is the value representing the enumeration session to be closed.
2182   The enumeration session shall be open and shall be established using one of the Open operations. This
2183   implies that this operation is not to close an enumeration sequence already indicated by
2184   `EndOfSequence` because the sequence has already been closed. The value of the
2185   `EnumerationContext` parameter shall be the value of the `EnumerationContext` output parameter
2186   returned by the previous Pull operation on the enumeration session to be closed.

2187   If CloseEnumeration is successful, the CIM server shall close the enumeration session represented by
2188   `EnumerationContext`, releasing any allocated resources. Any subsequent use of the
2189   `EnumerationContext` value is unsuccessful.

2190   CloseEnumeration may be executed concurrently with a Pull operation or an EnumerationCount operation
2191   on the same enumeration session. If a CIM server receives a CloseEnumeration operation request while
2192   it is processing a Pull operation on the same enumeration session, the server shall attempt to abandon
2193   that Pull operation. If the Pull operation can be abandoned, it shall return a failure with the status code
2194   <u>CIM_ERR_PULL_HAS_BEEN_ABANDONED</u> and the CloseEnumeration operation shall return success.

2195 If the Pull operation cannot be abandoned, it shall proceed as if the CloseEnumeration operation has not
2196 been issued, and the CloseEnumeration operation shall return a failure with the status code
2197 CIM_ERR_PULL_CANNOT_BE_ABANDONED.

2198 If CloseEnumeration is unsuccessful, this operation shall return one of the following status codes, where
2199 the error returned is the first applicable error in the list, starting with the first element and working down.
2200 Any additional operation-specific interpretation of the error is enclosed in parentheses.

2201 • CIM_ERR_ACCESS_DENIED

2202 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2203 • CIM_ERR_NOT_SUPPORTED

2204 • CIM_ERR_INVALID_NAMESPACE

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

2207 • CIM_ERR_INVALID_ENUMERATION_CONTEXT

2208 • CIM_ERR_PULL_CANNOT_BE_ABANDONED

2209 • CIM_ERR_FAILED (Some other unspecified error occurred.)

2210 **5.3.2.24.13 EnumerationCount**

2211 The EnumerationCount operation provides an estimated count of the total number of objects in an open
2212 enumeration session represented by an EnumerationContext:

```
2213     EnumerationCount
2214     uint64 EnumerationCount (
2215         [IN] <enumerationContext>EnumerationContext
2216     )
```

2217 The EnumerationContext parameter identifies the enumeration session for the EnumerationCount
2218 operation. It shall be established using any of the Open operations and shall be open at the time of the
2219 CloseEnumeration request. A conformant CIM server may support this operation. A CIM server that does
2220 not support this operation should respond with the CIM_ERR_NOT_SUPPORTED status.

2221 If EnumerationCount is successful, the operation returns an approximate count of the number of objects
2222 in the enumeration session. This is the number of items remaining to be sent with subsequent Pull
2223 operations. Thus, executing this operation immediately after the open may provide an approximate
2224 estimate of the total number of objects to be returned in the enumeration set. The returned count is only
2225 an estimate of the number of objects to be pulled in the enumeration sequence. This mechanism is
2226 intended to assist CIM clients in determining the overall size of an enumeration set and the number of
2227 objects remaining in the enumeration session. It should not be used instead of the EndOfSequence
2228 parameter to determine the end of an enumeration sequence.

2229 If the CIM server cannot or will not return an estimate of the number of objects to be returned for the
2230 enumerationContext, it may return success and the NULL value.

2231 If EnumerationCount is unsuccessful, this operation shall return one of the following status codes, where
2232 the error returned is the first applicable error in the list, starting with the first element and working down.
2233 Any additional operation-specific interpretation of the error is enclosed in parentheses.

2234 • CIM_ERR_ACCESS_DENIED

2235 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2236 • CIM_ERR_NOT_SUPPORTED

2237 • CIM_ERR_INVALID_NAMESPACE

2238 2239	• CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise incorrect parameters)
2240	• CIM_ERR_INVALID_ENUMERATION_CONTEXT
2241	• CIM_ERR_SERVER_LIMITS_EXCEEDED
2242	• CIM_ERR_FAILED (Some other unspecified error occurred.)

2243   **5.3.2.24.14 OpenQueryInstances**

2244   The OpenQueryInstances operation establishes and opens an enumeration session of the instances of a
2245   CIM class (including instances of its subclasses) in the target namespace. Optionally, it retrieves a first
2246   set of instances:

```
2247        OpenQueryInstances
2248        <instance>* OpenQueryInstances (
2249            [IN] string FilterQuery,
2250            [IN] string FilterQueryLanguage,
2251            [IN,OPTIONAL] Boolean ReturnQueryResultClass = false,
2252            [IN,OPTIONAL,NULL] uint32 OperationTimeout = NULL,
2253            [IN,OPTIONAL] Boolean ContinueOnError = false,
2254            [IN,OPTIONAL] uint32 MaxObjectCount = 0,
2255            [OUT, OPTIONAL, NULL] <class> QueryResultClass,
2256            [OUT] <enumerationContext> EnumerationContext,
2257            [OUT] Boolean EndOfSequence
2258        )
```

2259   The OpenQueryInstances shall comply with the behavior defined in 5.3.2.24.1.

2260   The `FilterQuery` and `FilterQueryLanguage` input parameters specify the set of enumerated
2261   instances.

2262   `FilterQueryLanguage` shall specify a query language and the value of `FilterQuery` shall be a valid
2263   query in that query language. This specification defines neither the query language nor the format of the
2264   query. It is anticipated that query languages will be submitted to the DMTF as separate proposals. A
2265   mechanism by which CIM servers can declare the query languages they support for filtering in Pulled
2266   enumerations (if any) is defined in 7.5.

2267   The `ReturnQueryResultClass` input parameter controls whether a class definition is returned in
2268   `QueryResultClass`. If it is set to `false`, `QueryResultClass` shall be set to NULL on output. If it is
2269   set to `true`, the value of the `QueryResultClass` on output shall be a class definition that defines the
2270   properties (columns) of each row of the query result.

2271   The `OperationTimeout` input parameter is defined in 5.3.2.24.2.

2272   The `ContinueOnError` input parameter is defined in 5.3.2.24.2.

2273   The `MaxObjectCount` input parameter is defined in 5.3.2.24.2.

2274   The `QueryResultClass` output parameter shall be set to NULL if the `ReturnQueryResultClass`
2275   input parameter is set to `false`. Otherwise, it shall return a class definition where each property of the
2276   class corresponds to one entry of the query select list. The class definition corresponds to one row of the
2277   query result. The class name of this returned class shall be "CIM_QueryResult." This class definition is
2278   valid only in the context of this enumeration.

2279   The `EnumerationContext` output parameter is defined in 5.3.2.24.2.

2280   The `EndOfSequence` output parameter is defined in 5.3.2.24.2.

2281 If OpenQueryInstances is successful, the return value shall be an array of enumerated instances as
2282 defined in 5.3.2.24.2. Such instances are available only in the context of the enumeration and do not
2283 return an instance path. The PullInstancesWithPath operation may not be used to continue an
2284 enumeration started by the OpenQueryInstances operation.

2285 The PullInstances operation shall be used to pull instances for an enumeration session opened using If
2286 OpenQueryInstances. If any other operation is used to pull instances, the CIM server shall return failure
2287 with the status code CIM_ERR_FAILED.

2288 If OpenQueryInstances is unsuccessful, this operation shall return one of the following status codes,
2289 where the error returned is the first applicable error in the list, starting with the first element and working
2290 down. Any additional operation-specific interpretation of the error is enclosed in parentheses.

2291 • CIM_ERR_ACCESS_DENIED

2292 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2293 • CIM_ERR_NOT_SUPPORTED

2294 • CIM_ERR_INVALID_NAMESPACE

2295 • CIM_ERR_INVALID_OPERATION_TIMEOUT

2296 • CIM_ERR_CONTINUATION_ON_ERROR_NOT_SUPPORTED

2297 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2298   incorrect parameters)

2299 • CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED (The requested filter query language is
2300   not recognized.)

2301 • CIM_ERR_INVALID_QUERY (The filter query is not a valid query in the specified filter query
2302   language.)

2303 • CIM_ERR_QUERY_FEATURE_NOT_SUPPORTED (The query requires support for features
2304   that are not supported.)

2305 • CIM_ERR_FAILED (Some other unspecified error occurred.)

2306 **5.3.2.24.15 PullInstances**

2307 The PullInstances operation retrieves instances from an OpenQueryInstances session represented by an
2308 enumeration context value:

```
2309     PullInstances
2310     <instance>* PullInstances (
2311         [IN,OUT] <enumerationContext> EnumerationContext,
2312         [OUT] Boolean EndOfSequence,
2313         [IN] uint32 MaxObjectCount
2314     )
```

2315 The PullInstances operation shall comply with the behavior defined in 5.3.2.24.1.

2316 The `EnumerationContext` input/output parameter is defined in 5.3.2.24.9. The enumeration session
2317 shall be established using the OpenQueryInstances operation.

2318 The `EndOfSequence` output parameter is defined in 5.3.2.24.9.

2319 The `MaxObjectCount` input parameter is defined in 5.3.2.24.9.

2320 If PullInstances is successful, the return value shall be an array of enumerated instances as defined in
2321 5.3.2.24.9.

2322 If PullInstances is unsuccessful, this operation shall return one of the following status codes, where the
2323 error returned is the first applicable error in the list, starting with the first element and working down. Any
2324 additional operation-specific interpretation of the error is enclosed in parentheses.

2325 • CIM_ERR_ACCESS_DENIED

2326 • CIM_ERR_SERVER_IS_SHUTTING_DOWN

2327 • CIM_ERR_NOT_SUPPORTED

2328 • CIM_ERR_INVALID_NAMESPACE

2329 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2330 incorrect parameters)

2331 • CIM_ERR_INVALID_ENUMERATION_CONTEXT

2332 • CIM_ERR_SERVER_LIMITS_EXCEEDED

2333 • CIM_ERR_PULL_HAS_BEEN_ABANDONED

2334 • CIM_ERR_FAILED (Some other unspecified error occurred.)

2335 5.3.3 Namespace Manipulation Using the CIM_Namespace Class

2336 No intrinsic methods are defined specifically to manipulate namespaces. Namespaces shall be
2337 manipulated using intrinsic methods on the CIM_Namespace class.

2338 5.3.3.1 Namespace Creation

2339 A namespace is created by calling the intrinsic method CreateInstance for the CIM_Namespace class. A
2340 value is specified for the new instance parameter that defines a valid instance of the CIM_Namespace
2341 class and that has a name property that is the desired name of the new namespace.

2342 The proposed definition shall be a correct namespace definition according to DSP0004. Despite the
2343 naming conventions used in the CIM specifications (use of / in namespaces such as root/CIMV2 and
2344 root/CIMV2/test), there is no hierarchy implied among different namespaces. Each namespace is
2345 independent of all others. The namespaces are to be considered flat, and there is no defined behavior for
2346 navigating namespaces.

2347 In creating the new namespace, the CIM server shall conform to the following rules:

2348 • The namespace defined by name property shall not already exist in the CIM server.

2349 • The <LOCALNAMESPACEPATH> defined for the operation defines the namespace in which
2350 the CIM_Namespace instance associated with this new namespace is created.

2351 It is recommended that instances of CIM_Namespace be created in root unless there is a specific reason
2352 to define them in another namespace. The inclusion of a CIM_Namespace instance within a namespace
2353 other than root is allowed.

2354 In addition to creating instances of CIM_Namespace, compliant implementations shall also create an
2355 instance of the association class CIM_NamespaceInManager defining the linking of the namespace
2356 created to the current CIM_ObjectManager.

2357 If CreateInstance is successful, the CIM server creates the specified namespace. In addition, the CIM
2358 server shall return information about the namespace as an instance of the class CIM_Namespace and of
2359 returning instances of the association class CIM_NamespaceInManager for each CIM_Namespace
2360 instance created.

2361 **5.3.3.2 Namespace Deletion**

2362 If the CIM server supports the CIM_Namespace class, all valid namespaces shall be represented by an
2363 instance of the CIM_Namespace class. A namespace is deleted using the intrinsic method
2364 DeleteInstance to delete the instance of the class CIM_Namespace that represents the namespace. The
2365 namespace to be deleted shall exist.

2366 If DeleteInstance is successful, the CIM server shall remove the specified CIM_Namespace instance.

2367 If DeleteInstance is unsuccessful, one of the status codes defined for the DeleteInstance operation shall
2368 be returned. A CIM server may return CIM_ERR_FAILED if a non-empty namespace cannot successfully
2369 be deleted.

2370 **5.3.3.3 Manipulation and Query of Namespace Information**

2371 The query of namespaces is provided through the following means:

2372 • Query of the CIM_Namespace class on an individual namespace

2373 • Use of the CIM_NamespaceInManager association to link the target CIM_ObjectManager and
2374 the instances of CIM_Namespace representing all namespaces defined in the target
2375 CIM_ObjectManager

2376 **5.3.3.4 Use of the __Namespace Pseudo Class (DEPRECATED)**

2377 In previous versions of this specification, namespaces were manipulated through the pseudo class
2378 __Namespace as follows:

2379 No intrinsic methods are specifically defined for manipulating CIM namespaces. However, modeling a
2380 CIM namespace using class __Namespace, together with the requirement that the root namespace be
2381 supported by all CIM servers, implies that all namespace operations can be supported.

2382 For example, all child namespaces of a particular namespace are enumerated by calling the intrinsic
2383 method EnumerateInstanceNames against the parent namespace, specifying a value for the ClassName
2384 parameter of __Namespace. A child namespace is created by calling the intrinsic method CreateInstance
2385 against the parent namespace, specifying a value for the NewInstance parameter that defines a valid
2386 instance of the class __Namespace and that has a name property that is the desired name of the new
2387 namespace.

2388 **DEPRECATION NOTE:** The use of the __Namespace class is DEPRECATED. In its place, use the
2389 CIM_Namespace class.

2390 **5.3.4 Functional Profiles**

2391 To establish conformance, this clause partitions the intrinsic methods into functional groups.

2392 Support for a particular group does *not* guarantee that all invocations of a method in that group will
2393 succeed. Rather, the exclusion of a group is a declaration that any attempt to call a method in that group
2394 always returns CIM_ERR_NOT_SUPPORTED.

2395 Mechanisms by which a CIM server may declare the functional groups that it supports are defined in 7.5.

2396 To limit the number of different profiles that a CIM server may support, each functional group has a
2397 dependency on another group (with the exception of the Basic Read functional group). If functional group
2398 G_1 has a dependency on functional group G_2, then a CIM server that supports G_1 shall also support G_2.

2399 The dependency relation is transitive, so if G_1 depends on G_2, and G_2 depends on G_3, then G_1 depends
2400 on G_3. It is also anti-symmetric, so if G_1 depends on G_2, then G_2 cannot depend on G_1.

2401   Using these rules, Table 3 defines a rooted-directed tree of dependencies with the Basic Read
2402   dependency representing the root node.

2403   For example, a CIM server that supports the Schema Manipulation functional group shall also support the
2404   Instance Manipulation, Basic Write, and Basic Read.

2405   A CIM server shall support the Basic Read functional group.

2406   **Table 3 – Root-Directed Tree of Functional Profile Dependencies**

Functional Group	Dependency	Methods
Basic Read	*none*	GetClass EnumerateClasses EnumerateClassNames GetInstance EnumerateInstances EnumerateInstanceNames GetProperty
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
Schema Manipulation	Instance Manipulation	CreateClass ModifyClass DeleteClass
Instance Manipulation	Basic Write	CreateInstance ModifyInstance DeleteInstance
Association Traversal	Basic Read	Associators AssociatorNames References ReferenceNames
Query Execution	Basic Read	ExecQuery
Qualifier Declaration	Schema Manipulation	GetQualifier SetQualifier DeleteQualifier EnumerateQualifiers

5.3.5   Extrinsic Method Invocation

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

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

- CIM_ERR_ACCESS_DENIED

- CIM_ERR_NOT_SUPPORTED (The CIM server does not support extrinsic method invocations.)

- CIM_ERR_INVALID_NAMESPACE

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

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

- CIM_ERR_METHOD_NOT_FOUND

- CIM_ERR_METHOD_NOT_AVAILABLE (The CIM server is unable to honor the invocation request.)

- CIM_ERR_FAILED (Some other unspecified error occurred.)

5.4   CIM Export Syntax and Semantics

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

5.4.1   Export Method Invocations

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

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

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

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

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

- Input parameters may be supplied in any order.

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

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

2447 • Each output parameter shall be named using the name assigned in the method definition.

2448 • Output parameters may be supplied in any order.

2449 • Each output parameter of the method, and no others, shall be present in the response, unless it
2450   is optional.

2451 The method invocation process may be thought of as a two-part process:

2452 • Binding the input parameter values specified as subelements of the <EXPMETHODCALL>
2453   element to the input parameters of the method.

2454 • Attempting to execute the method using the bound input parameters, with one of the following
2455   results:

2456   – If the attempt to call the method is successful, the return value and output parameters are
2457   bound to the subelements of the <EXPMETHODRESPONSE> element.

2458   – If the attempt to call the method is unsuccessful, an error code and (optional) human-
2459   readable description of that code is bound to the <EXPMETHODRESPONSE> element.

5.4.1.1 Simple Export

2460

2461 A simple export requires the invocation of a single export method. A simple export request is represented
2462 by a <SIMPLEEXPREQ> element, and a simple export response is represented by a <SIMPLEEXPRSP>
2463 element.

2464 A <SIMPLEEXPREQ> shall contain a <EXPMETHODCALL> element.

5.4.1.2 Multiple Export

2465

2466 A multiple export requires the invocation of more than one export method. A multiple export request is
2467 represented by a <MULTIEXPREQ> element, and a multiple export response is represented by a
2468 <MULTIEXPRSP> element.

2469 A <MULTIEXPREQ> (or its respective <MULTIEXPRSP>) element is a sequence of two or more
2470 <SIMPLEEXPREQ> (or its respective <SIMPLEEXPRSP>) elements.

2471 A <MULTIEXPRSP> element shall contain a <SIMPLEEXPRSP> element for every <SIMPLEEXPREQ>
2472 element in the corresponding multiple export response. These <SIMPLEEXPRSP> elements shall be in
2473 the same order as their <SIMPLEEXPREQ> counterparts. The first <SIMPLEEXPRSP> in the response
2474 corresponds to the first <SIMPLEEXPREQ> in the request, and so forth.

2475 Multiple exports conveniently batch the delivery of multiple export method invocations into a single HTTP
2476 message, reducing the number of roundtrips between a CIM client and a CIM listener and allowing the
2477 CIM listener to make certain internal optimizations. Note that multiple exports do not confer any
2478 transactional capabilities in processing the request. For example, the CIM listener does not have to
2479 guarantee that the constituent export method calls either all failed or all succeeded. The CIM listener
2480 must only make a "best effort" to process the operation. However, CIM listeners shall finish processing
2481 each method invocation in a batched message before executing the next method invocation in the batch.
2482 Clients shall recognize that the order of method calls within a batched message is significant.

2483 Not all CIM listeners support multiple exports. If a CIM listener does not support multiple exports, it shall
2484 return the status code CIM_ERR_NOT_SUPPORTED.

5.4.1.3 Status Codes

2485

2486 This clause defines the status codes and detailed error information that a conforming CIM listener may
2487 return.

2488 The value of an <ERROR> subelement within a <EXPMETHODRESPONSE> element includes the
2489 following parts:

2490 • mandatory status code

2491 • optional human-readable description of the status code

2492 • zero or more CIM_Error instances

2493 The symbolic names defined in Table 4 do not appear on the wire. They are used here solely for
2494 convenient reference to an error in other parts of this specification. Not all methods are expected to return
2495 all these status codes.

2496 In addition to returning a status code, a conforming CIM listener may return zero or more <INSTANCE>
2497 subelements as part of an <ERROR> element. Each <INSTANCE> subelement shall be an instance of
2498 CIM_Error, and the value of CIMStatusCode shall comply with the definition of expected error codes for
2499 the CIM export request. A CIM client may ignore any <INSTANCE> subelements.

2500 **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.

2501 ## 5.4.2 Export Methods

2502 This clause describes the methods that can be defined within a CIM export message. These methods
2503 operate only on an external data representation of a CIM entity, namespace, or element. Specifically,
2504 export methods do not operate on CIM namespaces or elements. The export method defined in this
2505 specification is Export an Indication.

2506 The notation used in the following subclauses to define the signatures of the export methods is a pseudo-
2507 MOF notation that extends the standard MOF BNF (DSP0004) for describing CIM export methods with a
2508 number of pseudo parameter types. The pseudo parameter types are enclosed in angle brackets (< >).

2509 This notation allows parameters to be decorated with pseudo-qualifiers (IN, OPTIONAL, and NULL) to
2510 define their invocation semantics. Note that these qualifiers are for description purposes only within the
2511 scope of this specification. In particular, a CIM client shall not specify them in export method invocations.

2512 This notation uses the IN qualifier for input parameters.

2513 A CIM client may omit an optional parameter if the required value is the specified default by not specifying
2514 an <EXPPARAMVALUE> element for the parameter. It shall not omit a parameter that is not optional.

2515 The NULL qualifier indicates parameters with values that may be specified as NULL in an export method
2516 call. A NULL (unassigned) value for a parameter is specified by an <EXPPARAMVALUE> element with
2517 no subelement. The CIM client shall specify a value for parameters without the NULL qualifier by
2518 including a suitable subelement for the <EXPPARAMVALUE> element.

2519 All parameters shall be uniquely named and shall correspond to a valid parameter name for that method
2520 as described by this specification. The order of the parameters is not significant.

2521 The non-NULL values of export method parameters or return values that are modeled as standard CIM
2522 types (such as string and Boolean, or arrays thereof) are represented as follows:

2523 • Simple values shall be represented by the <VALUE> subelement in an <EXPPARAMVALUE>
2524 element (for export method parameters) or in an <IRETURNVALUE> element (for export
2525 method return values).

2526 • Array values shall be represented by the <VALUE.ARRAY> subelement in an
2527 <EXPPARAMVALUE> element (for export method parameters) or in an <IRETURNVALUE>
2528 element (for export method return values).

2529 Table 5 shows how each pseudo-type used by the export methods shall be mapped to an XML element
2530 described in DSP0201 in the context of both a parameter value (subelement of <EXPPARAMVALUE>)
2531 and a return value (subelement of <IRETURNVALUE>).

2532 **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

2533 **5.4.2.1 ExportIndication**

2534 The ExportIndication operation exports a single CIM indication to the destination CIM listener:

```
2535     ExportIndication
2536     void ExportIndication (
2537         [IN] <instance> NewIndication
2538     )
```

2539 The `NewIndication` input parameter defines the indication to be exported. The proposed definition
2540 should be a correct instance definition for the underlying CIM indication class according to the CIM
2541 specification.

2542 If ExportIndication is unsuccessful, this method shall return one of the following status codes, where the
2543 error returned is the first applicable error in the list, starting with the first element and working down. Any
2544 additional method-specific interpretation of the error is enclosed in parentheses.

2545 • CIM_ERR_ACCESS_DENIED

2546 • CIM_ERR_NOT_SUPPORTED

2547 • CIM_ERR_INVALID_PARAMETER (including missing, duplicate, unrecognized, or otherwise
2548 incorrect parameters)

2549 • CIM_ERR_INVALID_CLASS (The CIM class of which this is to be a new instance does not
2550 exist.)

2551 • CIM_ERR_FAILED (Some other unspecified error occurred.)

2552 **5.4.3 Functional Profiles**

2553 This clause partitions the export methods into functional groups to establish conformance. See Table 6.

2554 Support for a particular group does not guarantee that all invocations of an export method in that group
2555 will succeed. Rather, the exclusion of a group is a declaration that any attempt to call an export method in
2556 that group always returns CIM_ERR_NOT_SUPPORTED.

2557 The dependency relation is transitive, so if group G_1 depends on G_2, and G_2 depends on G_3, then G_1
2558 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.

2559

Table 6 – Functional Groups of Export Methods

Functional Group	Dependency	Method
Indication	None	ExportIndication

6   Encapsulation of CIM Messages

2561   This clause describes how to use CIM messages in HTTP. CIM message requests may be used with or
2562   without the HTTP Extension Framework.

2563   Although CIM messages can be used in combination with a variety of HTTP request methods, this
2564   specification defines CIM messages only within HTTP POST requests. (M-POST may be used in place of
2565   POST. For details on how to use CIM messages with the HTTP Extension Framework, see 6.2.)

2566   All CIM message responses are carried in the corresponding HTTP response. In the remaining
2567   discussion, the following terms are used as convenient shorthand for the definitions provided here:

2568   • *CIM operation request.* An HTTP POST request message with an XML entity body that defines
2569   an Operation Request Message.

2570   • *CIM operation response.* An HTTP response message, issued in response to a CIM operation
2571   request, with an entity body that defines an Operation Response Message.

2572   • *CIM export request.* An HTTP POST request message with an XML entity body that defines a
2573   CIM export message request.

2574   • *CIM export response.* An HTTP response message, issued in response to a CIM export
2575   message request, with an entity body that defines a CIM export message response.

2576   • *CIM message request.* An HTTP POST request message with an XML entity body that defines
2577   either a CIM operation or export message request.

2578   • *CIM message response.* An HTTP response message, issued in response to a CIM operation
2579   message (or CIM export message) request, with an entity body that defines a CIM operation
2580   message (or CIM export message) response.

2581   Note that an HTTP response to a CIM request is not always a CIM response. For example, a "505 HTTP
2582   Version Not Supported" response is not a CIM response.

6.1   CIM Clients, CIM Servers, and CIM Listeners

2584   A CIM product is any product that can supply and/or consume management information using the CIM
2585   schema. In particular, CIM clients, CIM servers, and CIM listeners are examples of CIM products:

2586   • A *CIM client* issues CIM operation requests (CIM message requests) and receives and
2587   processes CIM operation responses (CIM message responses).

2588   • A *CIM server* receives and processes CIM operation message requests and issues CIM
2589   operation message responses.

2590   • A *CIM listener* is a server that receives and processes CIM export message requests and
2591   issues CIM export message responses.

2592   A CIM server may be act as any combination of CIM client, CIM server, and CIM listener. For example, a
2593   CIM server that supports indication subscription and generation acts as a CIM client when delivering an
2594   indication, through ExportIndication, to a CIM listener.

2595   Throughout this document, the terms CIM client, CIM server, CIM listener, and CIM product are used as
2596   convenient shorthand to refer to the subset of CIM products that conform to this specification.

6.2   Use of M-POST

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

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

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

- For all other status codes, the client shall act in accordance with standard HTTP (RFC 1945, RFC 2068).

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

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

6.2.1   Use of the Ext Header

If a CIM server or CIM listener receives a valid M-POST request and has fulfilled all mandatory extension header declarations in the request, it shall include in the response the "Ext" header defined by RFC 2774. This included header shall be protected by the appropriate Cache-Control directive.

6.2.2   Naming of Extension Headers

In M-POST request messages (and their responses), CIM extension headers shall be declared using the name space prefix allotted by the "Man" extension header (in accordance with RFC 2774) that refers to the name space "http://www.dmtf.org/cim/mapping/http/v1.0". The full format of the "Man" header declaration for this specification is:

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

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

In accordance with RFC 2774, all POST request messages (and their responses) shall not include such a mandatory extension declaration. In POST request messages (and their responses), name space prefixes shall not be used.

EXAMPLE 1:

    Using M-POST:

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

EXAMPLE 2:

    Using POST:

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

2640   ...

6.3   Extension Headers Defined for CIM Message Requests and Responses

2642   A CIM message contains exactly one CIM operation request, CIM operation response, CIM export
2643   request, or CIM export response. This clause describes the extension headers to specify CIM message
2644   semantics in the HTTP header of a POST message.

2645   Any CIM Operation Request or CIM Operation Response shall, and only CIM operation requests and
2646   responses may, include the following CIM extension header:

2647   • CIMOperation

2648   Any CIM Operation Request shall, and only CIM operation requests may, include one and only one of the
2649   following CIM extension header sets:

2650   • CIMMethod and CIMObject, or

2651   • CIMBatch

2652   Any CIM export request or CIM export response shall, and only CIM export requests and responses may,
2653   include the following CIM extension header:

2654   • CIMExport

2655   Any CIM export request shall, and only CIM export requests may, include one and only one of the
2656   following CIM extension headers:

2657   • CIMExportMethod

2658   • CIMExportBatch

2659   An HTTP response with an error status code to a CIM message request may include the following CIM
2660   extension header:

2661   • CIMError

2662   All CIM messages may include the following CIM extension header:

2663   • CIMProtocolVersion

6.3.1   Encoding of CIM Element Names within HTTP Headers and Trailers

2665   CIM element (class, property, qualifier, method, or method parameter) names are natively Unicode, and
2666   may use UCS-2 characters unsuitable for inclusion within an HTTP message header or trailer. To encode
2667   CIM element names represented in Unicode to values within HTTP headers or trailers, the following two-
2668   step mapping process shall be used:

2669   • Encode the full Unicode CIM element name using UTF-8.

2670   • Using the ""%" HEX HEX" convention, apply the standard URI [RFC 2396, section 2] escaping
2671   mechanism to the resulting string to escape any characters that are unsafe within an HTTP
2672   header or trailer.

2673   In this specification, the token CIMIdentifier represents a CIM element name to which this transformation
2674   has been applied.

2675   One characteristic of this mapping is that CIM elements named with an ASCII representation appear in
2676   ASCII in the resulting URL.

2677 EXAMPLE:

2678 CIM_LogicalElement is unchanged under this transformation.

2679 The class named using the UCS-2 sequence representing the Hangul characters for the Korean word "hangugo"
2680 (D55C, AD6D, C5B4) becomes

2681 %ED%95%9C%EA%B5%AD%EC%96%B4=10

2682 after UTF-8 transformation and escaping all characters with their % HEX HEX equivalent.

2683 6.3.2 Encoding of CIM Object Paths within HTTP Headers and Trailers

2684 This clause describes the mapping that shall be applied to represent CIM object paths, as described
2685 within an Operation Request Message using the <LOCALNAMESPACEPATH>, <LOCALCLASSPATH>,
2686 or <LOCALINSTANCEPATH> elements, in a format that is safe for representation within an HTTP header
2687 or trailer.

2688 If the element to be transformed is a <LOCALNAMESPACEPATH>, the algorithm is as follows:

2689 • For the first NAMESPACE subelement, output the textual content of that element.

2690 • For each subsequent NAMESPACE subelement, output the forward slash character (/) followed
2691 by the textual content of that NAMESPACE element.

2692 If the element to be transformed is a <LOCALCLASSPATH>, the algorithm is as follows:

2693 • Transform the <LOCALNAMESPACEPATH> subelement using the rules previously described,
2694 and output a colon character (:).

2695 • Output the value of the NAME attribute of the <CLASSNAME> subelement.

2696 If the element to be transformed is a <LOCALINSTANCEPATH>, the algorithm is as follows:

2697 • Transform the <LOCALNAMESPACEPATH> subelement using the rules previously described,
2698 and output a colon character (:).

2699 • Output the value of the CLASSNAME attribute of the <INSTANCENAME> subelement.

2700 • If there is at least one <KEYBINDING> subelement under the <INSTANCENAME> subelement,
2701 then for each such subelement:

2702 – Output a period character (.) if this is the first <KEYBINDING> subelement; otherwise,
2703 output a comma character (,).

2704 – Output the value of the NAME attribute, followed by an equal character (=).

2705 – If there is a <KEYVALUE> subelement, output the textual element content of that element,
2706 subject to the following transformation:

2707 ▪ If the VALUETYPE attribute is numeric or Boolean, the output is identical to the content
2708 of the element.

2709 ▪ If the VALUETYPE attribute is a string, the output is obtained by enclosing the content
2710 of the element in double quote (") characters and escaping any double quote
2711 characters or backslash character within the value with a preceding backslash (\)
2712 character.

2713 – If there is a <VALUE.REFERENCE> subelement

2714 ▪ Output a double quote character (").

- ▪ Apply the process recursively to the <CLASSPATH> or <INSTANCEPATH> subelement of the <VALUE.REFERENCE> element, escaping any double quote or backslash character thereby generated with a preceding backslash (\) character.

  - ▪ Output a closing double quote character (").

- • If there is no <KEYBINDING> subelement but there is a <KEYVALUE> or <VALUE.REFERENCE> subelement under the <INSTANCENAME> subelement, then:

  - – Output an equal character (=).

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

- • If there are no <KEYBINDING> subelements or no <KEYVALUE> or <VALUE.REFERENCE> subelement, then indicate a singleton instance by outputting the string "=@" under the <INSTANCENAME> subelement.

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

- • Encode the string using UTF-8 [RFC 2279] if it is not already in this format.

- • Using the ""%" HEX HEX" convention, apply the standard URI [RFC 2396, section 2] escaping mechanism to the resulting string to escape any characters that are unsafe within an HTTP header or trailer.

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

6.3.3 CIMOperation

The CIMOperation header shall be present in all CIM Operation Request and CIM Operation Response messages. It identifies the HTTP message as carrying a CIM operation request or response.

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

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

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

If a CIM server receives a CIM operation request without this header, it shall not process it as a CIM operation request. The status code returned by the CIM server in response to such a request is outside the scope of this specification.

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

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

2756 **6.3.4 CIMExport**

2757 The CIMExport header shall be present in all CIM export request and response messages. It identifies the
2758 HTTP message as carrying a CIM export method request or response.

2759   CIMExport = "CIMExport" ":" ("MethodRequest" | "MethodResponse")

2760 A CIM client shall include this header with the value "MethodRequest" in all CIM export requests that it
2761 issues. A CIM listener shall include this header in all CIM export responses that it issues, with the value
2762 "MethodResponse".

2763 If a CIM listener receives a CIM export request with this header, but with a missing value or a value that is
2764 not "MethodRequest", then it shall fail the request with status "400 Bad Request". The CIM listener shall
2765 include a CIMError header in the response with a value of unsupported-operation.

2766 If a CIM listener receives a CIM export request without this header, it shall not process it. The status code
2767 returned by the CIM listener in response to such a request is outside of the scope of this specification.

2768 If a CIM client receives a response to a CIM export request without this header (or if this header has a
2769 value that is not "MethodResponse"), it should discard the response and take appropriate measures to
2770 publicize that it has received an incorrect response. The details as to how this is done are outside the
2771 scope of this specification.

2772 The CIMExport header affords a simple mechanism by which firewall or proxy administrators can make
2773 global administrative decisions on all CIM exports.

2774 **6.3.5 CIMProtocolVersion**

2775 The CIMProtocolVersion header may be present in any CIM message. The header identifies the version
2776 of the CIM operations over the HTTP specification in use by the sending entity.

2777   CIMProtocolVersion = "CIMProtocolVersion" ":" 1*DIGIT "." 1*DIGIT

2778 If the header is omitted, then a value of 1.0 must be assumed.

2779 The major and minor revision numbers must be treated as independent integers.

2780 The CIMProtocolVersion $x_1.y_1$ is less than CIMProtocolVersion $x_2.y_2$ if and only if one of the following
2781 statements is true:

2782   • x_1 is less than x_2

2783   • x_1 equals x_2, and y_1 is less than y_2

2784 The CIMProtocolVersion $x_1.y_1$ is greater than CIMProtocolVersion $x_2.y_2$ if and only if one of the following
2785 statements is true:

2786   • x_1 is greater than x_2,

2787   • x_1 equals x_2, and y_1 is greater than y_2

2788 A CIMProtocolVersion $x_1.y_1$ is within tolerance of CIMProtocolVersion $x_2.y_2$ if:

2789   • x_1 equals x_2, and

2790   • y_1 is less than or equal to y_2

2791 If the CIMProtocolVersion of the CIM message received is within tolerance of the CIMProtocolVersion
2792 supported for a CIM server or CIM listener implementation, the receiving implementation shall accept that
2793 CIM message. Equivalent CIMProtocolVersion values between CIM server or CIM listener and the CIM

2794   client shall be accepted. The CIM server or CIM listener implementation may reject a CIM message in all
2795   other cases. For information about how CIM messages are rejected, see 7.3.

2796   Beyond tolerance considerations, the implementation should reject the received CIM message *only* if the
2797   design as defined by the CIMProtocolVersion of the receiving implementation has changed in the
2798   declaration of the API, method parameters, or behavior since the design defined by the
2799   CIMProtocolVersion of the received CIM message.

6.3.6   CIMMethod

2801   The CIMMethod header shall be present in any CIM Operation Request message that contains a Simple
2802   Operation Request.

2803   It shall not be present in any CIM Operation Response message nor in any CIM Operation Request
2804   message unless it is a simple operation request. It shall not be present in any CIM export request or
2805   response message.

2806   The header identifies the name of the CIM method to be invoked, encoded in an HTTP-safe
2807   representation. Firewalls and proxies may use this header to carry out routing and forwarding decisions
2808   based on the CIM method to be invoked.

2809   The name of the CIM method within a simple operation request is the value of the NAME attribute of the
2810   <METHODCALL> or <IMETHODCALL> element.

2811   CIMMethod = "CIMMethod" ":" MethodName

2812   MethodName = CIMIdentifier

2813   If a CIM server receives a CIM operation request for which any one of the following statements is true,
2814   then it shall fail the request and return a status of "400 Bad Request". Also, it shall include a CIMError
2815   header in the response with a value of header-mismatch, subject to the considerations specified in 7.3:

2816   •   The CIMMethod header is present, but it has an invalid value.

2817   •   The CIMMethod header is not present, but the operation request message is a Simple
2818   Operation Request.

2819   •   The CIMMethod header is present, but the operation request message is not a simple operation
2820   request.

2821   •   The CIMMethod header is present and the operation request message is a simple operation
2822   request, but the CIMIdentifier value (when unencoded) does not match the unique method
2823   name within the simple operation request.

2824   Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy was
2825   not acting on misleading information when it decided to forward the request based on the content of the
2826   CIMMethod header. Additional securing of HTTP messages against modification in transit (such as the
2827   encryption of the payload or appending of a digital signature thereto) would be required to provide a
2828   higher degree of integrity.

6.3.7   CIMObject

2830   The CIMObject header shall be present in any CIM Operation Request message that contains a Simple
2831   Operation Request.

2832   It shall not be present in any CIM Operation Response message nor in any CIM Operation Request
2833   message unless it is a simple operation Request. It shall not be present in any CIM export request or
2834   response message.

2835 The header identifies the CIM object on which the method is to be invoked using a CIM object path
2836 encoded in an HTTP-safe representation. This object shall be a class or instance for an extrinsic method
2837 or a namespace for an intrinsic method. Firewalls and proxies may use this header to carry out routing
2838 and forwarding decisions based on the CIM object that is the target of a method invocation.

2839 CIMObject = "CIMObject" ":" ObjectPath

2840 ObjectPath = CIMObjectPath

2841 The ObjectPath value is constructed by applying the algorithm defined in 6.3.2 to either of the following
2842 subelements within the CIM operation request:

2843 • The <LOCALNAMESPACEPATH> subelement of the <IMETHODCALL> element.

2844 • The <LOCALCLASSPATH> or <LOCALINSTANCEPATH> subelement of the
2845 <METHODCALL> element.

2846 If a CIM server receives a CIM operation request for which any one of the following statements is true,
2847 then it shall fail the request and return a status of "400 Bad Request". Also, it shall include a CIMError
2848 header in the response with a value of `header-mismatch`, subject to the considerations specified in 7.3:

2849 • The CIMObject header is present, but it has an invalid value.

2850 • The CIMObject header is not present, but the operation request message is a Simple Operation
2851 Request.

2852 • The CIMObject header is present, but the operation request message is not a simple operation
2853 request.

2854 • The CIMObject header is present and the operation request message is a simple operation
2855 request, but the ObjectPath value does not match the operation request message (where a
2856 *match* is defined in 6.3.2).

2857 Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy is not
2858 acting on misleading information when it forwards the request based on the content of the CIMObject
2859 header. Additional securing of HTTP messages against modification in transit, such as encrypting the
2860 payload or appending a digital signature to it, would be required to provide a higher degree of integrity.

2861 ## 6.3.8   CIMExportMethod

2862 The CIMExportMethod header shall be present in any CIM export request message that contains a
2863 simple export request.

2864 This header shall not be present in any CIM export response message nor in any CIM export request
2865 message unless it is a simple export request. It shall not be present in any CIM operation request or
2866 response message.

2867 The CIMExportMethod header identifies the name of the CIM export method to be invoked, encoded in an
2868 HTTP-safe representation. Firewalls and proxies may use this header to carry out routing and forwarding
2869 decisions based on the CIM export method to be invoked.

2870 The name of the CIM export method within a simple export request is the value of the NAME attribute of
2871 the <EXPMETHODCALL> element.

2872 CIMExportMethod = "CIMExportMethod" ":" ExportMethodName

2873 ExportMethodName = CIMIdentifier

2874 If a CIM listener receives a CIM export request for which any one of the following statements is true, then
2875 it shall fail the request and return a status of "400 Bad Request". Also, it shall include a CIMError header
2876 in the response with a value of header-mismatch, subject to the considerations specified in 7.3:

- The CIMExportMethod header is present, but it has an invalid value.

- The CIMExportMethod header is not present, but the export request message is a simple export request.

- The CIMExportMethod header is present, but the export request message is not a simple export request.

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

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

6.3.9   CIMBatch

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

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

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

      CIMBatch = "CIMBatch" ":"

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

- The CIMBatch header is present, but it has an invalid value.

- The CIMBatch header is not present, but the operation request message is a multiple operation request.

- The CIMBatch header is present, but the operation request message is not a multiple operation request.

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

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

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

2918 **6.3.10 CIMExportBatch**

2919 The CIMExportBatch header shall be present in any CIM export request message that contains a multiple
2920 export request.

2921 It shall not be present in any CIM operation request or response message. Also, it shall not be present in
2922 any CIM export response message nor in any CIM export request message unless it is a multiple export
2923 request.

2924 The header identifies the encapsulated Export Request Message as containing multiple export method
2925 invocations. Firewalls and proxies may use this header to carry out routing and forwarding decisions for
2926 batched CIM Export method invocations.

2927   CIMExportBatch = "CIMExportBatch" ":"

2928 If a CIM listener receives a CIM export request for which any one of the following statements is true, then
2929 it must fail the request and return a status of "400 Bad Request". Also, it must include a CIMError header
2930 in the response with a value of header-mismatch, subject to the considerations specified in Errors:

2931   • The CIMExportBatch header is present, but it has an invalid value.

2932   • The CIMExportBatch header is not present, but the export request message is a multiple export
2933   request.

2934   • The CIMExportBatch header is present, but the export request message is not a multiple export
2935   request.

2936 Note that this verification provides a *basic* level of assurance that any intermediate firewall or proxy is not
2937 acting on misleading information when it forwards the request based on the content of the
2938 CIMExportBatch header. Additional securing of HTTP messages against modification in transit, such as
2939 encrypting the payload or appending a digital signature to it, would be required to provide a higher degree
2940 of integrity.

2941 If a CIM listener receives a CIM export request for which the CIMExportBatch header is present, but the
2942 CIM listener does not support multiple exports, then it shall fail the request and return a status of "501 Not
2943 Implemented". Firewalls or Proxies may also employ this mechanism to compel a CIM client to use simple
2944 rather than multiple export requests.

2945 A CIM client that receives a response of "501 Not Implemented" to a multiple export request should
2946 resubmit that request as a series of simple export requests.

2947 **6.3.11 CIMError**

2948 The CIMError header may be present in any HTTP response to a CIM message request that is not a CIM
2949 message response.

2950 It shall not be present in any CIM message response or in any CIM message request.

2951 The CIMError header provides further CIM-specific diagnostic information if the CIM server or
2952 CIM listener encounters a fundamental error during processing of the CIM operation request and is
2953 intended to assist clients to further disambiguate errors with the same HTTP status code:

```
2954        CIMError  = "CIMError" ":" cim-error
2955        cim-error = "unsupported-protocol-version" |
2956            "multiple-requests-unsupported" |
2957            "unsupported-cim-version" |
2958            "unsupported-dtd-version" |
2959            "request-not-valid" |
2960            "request-not-well-formed" |
```

2961   `"request-not-loosely-valid" |`
2962   `"header-mismatch" |`
2963   `"unsupported-operation"`

6.3.12 CIMRoleAuthenticate

2965 A CIM server may return a CIMRoleAuthenticate header as part of the 401 Unauthorized response along
2966 with the WWW-Authenticate header. The CIMRoleAuthenticate header must meet the challenge of
2967 indicating the CIM server policy on role credentials.

2968   challenge = "credentialrequired" | "credentialoptional" | "credentialnotrequired"

2969 • A challenge of *credentialrequired* indicates that the CIM server requires that a CIM client must
2970   present a credential if it seeks to assume a role.

2971 • A challenge of *credentialoptional* indicates that the credential is optional. If a credential is not
2972   sent, the CIM server allows the role assumption if it is permitted for the given user. However,
2973   certain operations that require the role credential may not succeed.

2974 • A challenge of *credentialnotrequired* indicates that no credential is required to assume the role.

2975 Absence of the CIMRoleAuthenticate header indicates that the CIM server does not support role
2976 assumption. A CIM client should handle each of these cases appropriately.

2977 The challenge does not contain any authorization scheme, realm, or other information. A CIM client
2978 should extract this information from the WWW-Authenticate header. This implies that for any given
2979 request, the role credentials should use the same scheme as those required for the user credentials.

2980 A CIM server allows role assumption to succeed only if the user is allowed to assume the role. Therefore,
2981 even if appropriate credentials are presented, role assumption can fail. If either the user authentication or
2982 role assumption fails, the entire authentication operation fails.

2983 To maintain backward compatibility, a CIM server that supports role assumption must allow user
2984 authentication even if no role is specified.

6.3.13 CIMRoleAuthorization

2986 The CIMRoleAuthorization header is supplied along with the normal authorization header that the CIM
2987 client populates to perform user authentication. If the CIM client needs to perform role assumption and
2988 the server challenge is credentialrequired, the CIMRoleAuthorization header must be supplied with the
2989 appropriate credentials. The credentials supplied as part of the CIMRoleAuthorization header must use
2990 the same scheme as those specified for the authorization header, as specified in RFC 2617. Therefore,
2991 both Basic and Digest authentication are possible for the role credential.

2992 If the CIM client wishes to assume a role but does not wish to supply role credentials for server challenge
2993 credentialoptional or credentialnotrequired, the CIMRoleAuthorization header must set the auth-scheme
2994 field as specified in RFC 2617 to be "role". The auth-param must contain the role name.

2995 A CIM server that supports roles must be capable of handling the presence of credentials in the
2996 CIMRoleAuthorization header (that is auth-scheme not set to "role") regardless of whether it is expecting
2997 credentials or not. It may choose to ignore these credentials.

6.3.14 CIMStatusCodeDescription

2999 If a CIM product includes the CIMStatusCode trailer, it may also include the CIMStatusCodeDescription
3000 trailer. The value of this trailer is a string describing the nature of the error. A CIM product shall not
3001 include this trailer if the CIMStatusCode trailer is not present.

3002 **6.3.15 WBEMServerResponseTime**

3003 The WBEMServerResponseTime header may be present in any CIM response message. If it is present,
3004 the header shall contain a measure, specified in microseconds, of the elapsed time required by the CIM
3005 server to process the request and create a response. Specifically, WBEMServerResponseTime describes
3006 the time elapsed since the CIM server received the CIM request message and the associated CIM
3007 response message was ready to send to the CIM client.

3008 WBEMServerResponseTime = "WBEMServerResponseTime" ":", where the response time must be
3009 representable as a 64-bit unsigned integer value. If the actual elapsed time exceeds the maximum
3010 representable value, then the maximum value shall be returned. If the actual elapsed time is less than 1
3011 microsecond, then a 0 shall be returned.

3012 Although a CIM client may ignore the WBEMServerResponseTime header, it shall allow this header to be
3013 included in a response.

7 HTTP Requirements and Usage

3014

3015 This clause describes HTTP support and the use of standard headers.

7.1 HTTP Support

3016

3017 It is recommended that CIM clients, CIM servers, and CIM listeners support HTTP/1.1. CIM clients, CIM
3018 servers, and CIM listeners may support HTTP/1.0. CIM clients, CIM servers, and CIM listeners shall not
3019 be limited to any version of HTTP earlier than 1.0.

3020 CIM products that use extension headers as defined in this specification shall conform to the
3021 requirements defined in RFC 2774 for their use.

7.2 Use of Standard Headers

3022

3023 Unless otherwise stated in this specification, CIM products shall comply with the requirements on the use
3024 of headers described in RFC 1945, RFC 2068. This clause defines only *additional* requirements on CIM
3025 products with respect to the use of standard HTTP headers (RFC 1945, RFC 2068) in a CIM message.

3026 Note that CIM products should not use headers defined in RFC 2068 but deprecated from RFC 2616 (for
3027 example, Public, Content-Base).

7.2.1 Accept

3028

3029 If a CIM client includes an Accept header in a request, it shall specify a value that allows the server to
3030 return an entity body of "text/xml" or "application/xml" in the response.

3031 A CIM server or CIM listener shall accept any value for this header stating that "text/xml" or
3032 "application/xml" is an acceptable type for a response entity. A CIM server or CIM listener should return
3033 "406 Not Acceptable" if the Accept header indicates that neither of these content types is acceptable.

3034 If a CIM server or CIM listener accepts a request to return an entity of a type other than "text/xml" or
3035 "application/xml", the nature of the response is outside the scope of this specification.

7.2.2 Accept-Charset

3036

3037 If a CIM client includes an Accept-Charset header in a request, it shall specify a value that allows the CIM
3038 server or CIM listener to return an entity body using the character set "UTF-8".

3039   A CIM server or CIM listener shall accept any value for this header asserting that "UTF-8" is an
3040   acceptable character set for a response entity. If the client does not provide an Accept-Charset, then
3041   "UTF-8" should be assumed by the CIM server or CIM listener.

3042   Accept-Charset: UTF-8

3043   A CIM server or CIM listener shall return "406 Not Acceptable" if the character set requested in the
3044   Accept-Charset header is not supported.

3045   If a CIM server or CIM listener accepts a request to return an entity using a character set other than
3046   "UTF-8", the behavior of the subsequent CIM client and CIM server interaction is outside the scope of this
3047   specification. See 7.8 for details.

3048   7.2.3   Accept-Encoding

3049   If a CIM client includes an Accept-Encoding header in a request, it shall specify a q value that allows the
3050   CIM server or CIM listener to use the "Identity" encoding. The value shall be greater than 0 or not
3051   specified.

3052   Accept-Encoding: Identity
3053   Accept-Encoding: Identity; q=1.0

3054   A CIM server or CIM listener shall accept any value for this header asserting that "Identity" is an
3055   acceptable encoding for the response entity.

3056   A CIM server or CIM listener shall return "406 Not Acceptable" if the Accept-Encoding header indicates
3057   that the requested encoding is not acceptable.

3058   7.2.4   Accept-Language

3059   If a CIM client includes an Accept-Language header in a request, it shall request a language-range,
3060   special-range, or both. The CIM client shall also allow any language to be returned if the requested
3061   languages cannot be supported. This is accomplished by including the special-range, "*". The CIM client
3062   may request multiple languages. Each language has equal priority, unless a q value is provided.

3063   Accept-Language: zh, *
3064   Accept-Language: zh;q=1.0, en;q=.7, *

3065   Each CIM element in the response should be localized in only one language. A CIM element shall not be
3066   duplicated in the response because it is localized in more than one language.

3067   CIM servers may support multiple languages. A CIM product shall interpret the use of the special-range
3068   value, "*", as a request to return the response content using the default language defined for the target
3069   processing the request. Multiple targets, with different default language settings, may participate in the
3070   construction of a response. (See RFC 2616 section 3.10 and ISO 639-1.)

3071   See 7.8 for more information.

3072   7.2.5   Accept-Ranges

3073   CIM clients shall not include the Accept-Ranges header in a request. A CIM server or CIM listener shall
3074   reject a request that includes an Accept-Range header with a status of "406 Not Acceptable".

3075   7.2.6   Allow

3076   If a CIM server or CIM listener is returning a "405 Method Not Allowed" response to a CIM message
3077   request, then the Allow header shall include either M-POST or POST. Whether it includes any other
3078   HTTP methods is outside the scope of this specification.

3079 **7.2.7 Authorization**

3080 See 7.4 for details.

3081 **7.2.8 Cache-Control**

3082 Generally, a CIM message request may consist of a mixture of CIM method invocations, some of which
3083 may be eminently able to cache (for example, the manufacturer label on a disk drive) and some of which
3084 may be decidedly impossible to cache (for example, format a disk drive).

3085 Furthermore, the encapsulation of such multiple method invocations in an HTTP POST or M-POST
3086 means that if a CIM message request has any effect on an HTTP cache it is likely to be one of
3087 invalidating cached responses for the target CIM server or CIM listener. Indeed, HTTP/1.1 stipulates that
3088 by default POST responses cannot be cached unless the server indicates otherwise using an appropriate
3089 Cache-Control or Expires header.

3090 For these reasons, CIM message responses should not be considered as able to be cached. A CIM
3091 server or CIM listener should not include a Cache-Control header in a CIM message response that might
3092 indicate to a cache that the response can be cached.

3093 If the CIM server or CIM listener is responding to a CIM message request conveyed in an M-POST
3094 request, then in accordance with RFC 2774 the CIM server or CIM listener shall include a no-cache
3095 control directive to prevent inadvertent caching of the "Ext" header, as in the following example:

3096 EXAMPLE

```
3097    HTTP/1.1 200 OK
3098    Ext:
3099    Cache-Control: no-cache
3100    ...
```

3101 **7.2.9 Connection**

3102 The following courses of action are recommended for connections:

3103 • CIM clients should avoid the use of the "Connection: close" header unless it is known in
3104   advance that this is the only request likely to be sent out on that connection.

3105 • CIM servers and CIM listener support persistent connections wherever possible.

3106 Timeout mechanisms should be employed to remove idle connections on the CIM client, CIM server, and
3107 CIM listener. The details of timeout mechanisms are outside the scope of this specification. Clients should
3108 be cautious in retrying requests, especially if they are not idempotent (for example, method invocation).

3109 CIM clients, CIM servers, and CIM listeners should support pipelining (see RFC 2068, section 1.1.2.2) if
3110 possible, but be aware of the requirements defined in RFC 2068. In particular, attention is drawn to the
3111 requirement from RFC 2068 that clients not pipeline requests using non-idempotent methods or non-
3112 idempotent sequences of methods. A client that needs to send a non-idempotent request should wait to
3113 send that request until it receives the response status for the previous request.

3114 **7.2.10 Content-Encoding**

3115 If a CIM client includes a Content-Encoding header in a request, it should specify a value of "identity",
3116 unless there is good reason to believe that the server or listener can accept another encoding.

3117 **7.2.11 Content-Language**

3118 The Content-Language entity-header field of a CIM message describes the natural language(s) of the
3119 intended audience of the content.

3120   A CIM message may contain a Content-Language header. The value of the Content-Language header in
3121   a CIM response message shall be consistent with the Accept-Language values specified in the
3122   corresponding CIM request message. If the CIM server cannot determine one or more of the content
3123   languages used to construct the response, then the Content-Language entity shall not be returned.

3124   Multiple targets using different Content-Language values may participate in constructing a response. The
3125   Content-Language field shall reflect all Content-Language values used to construct the response. The
3126   content of a CIM message may contain elements in languages not listed in the Content-Language field.

3127   ```
        Content-Language: en
        ```

3128   See 7.8 for details.

7.2.12 Content-Range

3130   CIM clients, CIM servers, and CIM listeners shall not use this header.

7.2.13 Content-Type

3132   CIM clients, CIM servers, and CIM listeners shall specify (and accept) a media type for the Content-Type
3133   header of either "text/xml" or "application/xml" as defined in RFC 2376. In addition, they may specify and
3134   shall accept a "charset" parameter as defined in RFC 2616. If a "charset" parameter is specified, it shall
3135   have the value "utf-8" either with or without surrounding double quotes. The sending side should use
3136   the form without double quotes. The receiving side shall support both forms. If a "charset" parameter is
3137   not specified, the receiving side shall assume "utf-8" as a default.

3138   Examples of valid Content-Type headers are:

3139   ```
        Content-type: text/xml
3140   Content-type: text/xml; charset=utf-8
3141   Content-type: text/xml; charset="utf-8"
3142   Content-type: application/xml
3143   Content-type: application/xml; charset=utf-8
3144   Content-type: application/xml; charset="utf-8"
        ```

### 7.2.14  Expires

3146    For the reasons described in 7.2.8, a CIM server or CIM listener shall not include an Expires header in a
3147    CIM message response that might indicate to a cache that the response can be cached.

### 7.2.15  If-Range

3149    CIM clients, CIM servers, and CIM listeners shall not use this header.

### 7.2.16  Proxy-Authenticate

3151    See 7.4 for details.

### 7.2.17  Range

3153    CIM clients, CIM servers, and CIM listeners shall not use this header.

### 7.2.18  WWW-Authenticate

3155    See 7.4 for details.

## 7.3    Errors and Status Codes

This clause defines how CIM servers and CIM listeners shall handle errors that occur in processing a CIM message request. This specification 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 CIM server or CIM listener shall take appropriate action as dictated by its conformance to the relevant version of HTTP (RFC 1945, RFC 2068).

Otherwise, if there are any mandatory extension declarations that the server does not support it shall respond with a "510 Not Extended" status according to RFC 2774.

Otherwise, the request shall be processed in accordance with the relevant version of HTTP (RFC 1945, RFC 2068) and the additional rules defined in this document.

Assuming that the HTTP request is otherwise correct, the CIM server or CIM 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 CIM server or CIM listener. The CIM server or CIM listener shall include a CIMError header in the response with a value of `unsupported-protocol-version.`

    – The client specified a Multiple Operation Request (or multiple Export Request), and the CIM server (or CIM listener) does not support such requests. The CIM server or CIM 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 CIM server or CIM 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 CIM server or CIM listener shall include a CIMError header in the response with a value of `unsupported-dtd-version.`

- 401 Unauthorized

    The CIM server or CIM listener is configured to require that a client authenticate itself before it can issue CIM message requests to the server or listener.

- 403 Forbidden

    The CIM server or CIM listener does not allow the client to issue CIM message requests. The CIM server or CIM listener may alternatively respond with a "404 Not Found" if it does not wish to reveal this information to the client.

- 407 Proxy Authentication Required

    The CIM server or CIM listener is configured to require that the proxy authenticate itself before it can issue CIM message requests on behalf of a CIM client to the server or listener.

3198    Assuming that the CIM extension headers are correct, a validating CIM server or CIM listener (one that
3199    enforces the validity of the CIM message request with respect to the CIM XML DTD) shall use the
3200    following status code when processing the entity body containing the CIM message request:

3201    • 400 Bad Request

3202        The entity body defining the CIM message request is not well-formed or not valid with respect to
3203        the CIM XML DTD. The CIM server or CIM listener shall include a CIMError header in the
3204        response with a value of `request-not-well-formed` or `request-not-valid` (as
3205        appropriate).

3206    A loosely-validating CIM server or CIM listener only enforces the CIM message request to be loosely
3207    valid. Therefore, it may reject a CIM message request that is not loosely valid with an HTTP status code
3208    of 400 (Bad Request) before further processing. In this case, the CIM server or CIM listener shall include
3209    a CIMError header in the response with a value of `request-not-loosely-valid`.

3210    A loosely-validating CIM server or CIM listener shall reject a CIM message request that is not well-formed
3211    with an HTTP status code of 400 (Bad Request). In this case, the CIM server or CIM listener shall include
3212    a CIMError header in the response with a value of `request-not-well-formed`.

3213    A loosely-validating CIM server or CIM listener shall not reject an invalid CIM message request that is
3214    loosely valid in the XML sense.

3215    A loosely-validating CIM server or CIM listener shall ultimately signal an error to the CIM client if the CIM
3216    message request is not loosely valid. That is, the request is missing required content or the required
3217    content is incorrect, such as an attribute with an invalid value according to the CIM XML DTD. It is not
3218    mandated to reject a CIM message request before processing, for to do otherwise would compel the
3219    server or listener to check the complete request before processing can begin and this would be as
3220    expensive as requiring the server or listener to fully validate the request. Therefore, a loosely-validating
3221    server or listener may elect to begin processing the request and issuing a response (with an HTTP
3222    success status code) before verifying that the entire request is loosely valid.

3223    A CIM client may use the CIMValidation header mechanism to determine whether a CIM server or CIM
3224    listener is validating or loosely-validating.

3225    Assuming that the CIM message request is correctly formed as previously described, the CIM server or
3226    CIM listener shall process the request accordingly and return a CIM message response.

3227    The entity body shall be a correct CIM message response for that request.

3228    If the CIM message response contains an entity that is a simple message response, then the response
3229    status shall be "200 OK". Otherwise, the response status shall be "207 Multistatus".

3230    ## 7.4    Security Considerations

3231    CIM clients, CIM servers, and CIM listeners may elect not to use authentication, but only in environments
3232    where lack of security is not an issue.

3233    Basic authentication is described in RFC 1945 and RFC 2068. Digest authentication is defined in
3234    RFC 2069. Both authentication schemes are covered in a consolidated document (RFC 2617), which also
3235    makes a number of improvements to the original specification of digest authentication.

3236    Basic authentication provides a very rudimentary level of authentication, with the major weakness that the
3237    client password is sent over the wire in unencrypted form.

3238    For this reason, CIM clients, CIM servers, and CIM listeners shall not use basic authentication other than
3239    in the context of a highly secure environment (for example, in conjunction with SSL or in a physically
3240    secure private network). CIM servers and CIM listeners shall not send basic authentication credentials in
3241    a WWW-Authenticate header other than in the context of a highly secure environment.

3242 Conforming applications should support the digest authentication scheme. Because digest authentication
3243 verifies that both parties share a common secret without having to send that secret in the clear, it is more
3244 secure than basic authentication. However, CIM clients, CIM servers, and CIM listeners that require more
3245 robust protection should use encryption mechanisms such as SSL or SHTTP.

3246 CIM clients, CIM servers, and CIM listeners using basic or digest authentication shall comply with the
3247 requirements set forth in RFC 1945, RFC 2068, RFC 2069, and RFC 2617. This specification describes
3248 only *additional* requirements on CIM clients, CIM servers, and CIM listeners when these authentication
3249 schemes are used.

3250 CIM servers and CIM listeners should require that CIM clients authenticate themselves. This specification
3251 does not mandate this because it is recognized that in some circumstances the CIM server or CIM
3252 listener may not require or wish the overhead of employing authentication. CIM servers and CIM listeners
3253 should carefully consider the performance/security tradeoffs in determining how often to issue challenges
3254 to CIM clients.

3255 A CIM server or CIM listener that returns a "401 Unauthorized" response to a CIM message request
3256 should include in the WWW-Authenticate response-header either the "Basic" or "Digest" authentication
3257 values (but not both). This specification does not mandate use of basic or digest authentication because it
3258 is recognized that in some circumstances the CIM server or CIM listener may use bespoke authentication
3259 mechanisms not covered by RFC 2617. Similar considerations apply to the use of the Proxy-Authorization
3260 header in "407 Proxy Authentication Required".

## 3261  7.5    Determining CIM Server Capabilities

3262 If a CIM server can return capabilities information, there are two techniques for returning this information
3263 as defined in this specification:

3264 • The preferred technique is through the use of the classes defined in 7.5.1.

3265 • Alternatively, use of the HTTP OPTIONS method as defined in 7.5.2 is allowed because historically it
3266 is the original technique defined for requesting capabilities information.

3267 Use of the CIM classes defined in 7.5.1 is strongly encouraged and it is expected that this method will be
3268 enhanced and extended in the future to provide more capabilities information. The future use of the HTTP
3269 OPTIONS method to determine capabilities of CIM servers is discouraged. It will probably not be
3270 expanded significantly and may be reviewed for possible deprecation in the next major revision of this
3271 specification.

### 3272  7.5.1   Determining CIM Server Capabilities through CIM Classes

3273 A set of CIM classes is defined specifically to return CIM server capabilities information as follows:

3274 • CIM_ObjectManager

3275 This class is a type of CIM_Service that defines the capabilities of the target CIM server.

3276 • CIM_ObjectManagerCommunicationMechanism

3277 This class describes access to the target CIM server. It defines the capabilities of the CIM
3278 server that are available through the target Object Manager Communication mechanism. A CIM
3279 server is allowed to support different capabilities through different communication mechanisms.

3280 • CIM_CIMXMLCommunicationMechanism

3281 This class specializes on ObjectManagerCommunicationMechanism, adding properties specific
3282 to the CIM-XML encoding and protocol.

3283 • CIM_CommMechanismForManager

3284          This association between CIM_ObjectManager and
3285          CIM_ObjectManagerCommunicationMechanism defines the communications protocols (and
3286          corresponding capabilities) available on the target CIM server through the
3287          ObjectManagerCommunicationMechanism instances.

3288   A CIM client may use instances of these CIM classes to determine the CIM capabilities (if any) of the
3289   target CIM server. A CIM server that supports capabilities determination through these classes shall
3290   support at least the Enumerate Instance and Get Instance operations for the classes. The use of other
3291   methods of the basic read profile is optional. A CIM server that does not support the determination of CIM
3292   capabilities through these classes shall return CIM_ERR_NOT_FOUND to any instance or class request
3293   on these classes. These classes shall not be used for reporting any other information than capabilities of
3294   the target CIM server.

3295   To provide interoperability, the CIM object manager classes shall exist in a well-known namespace.
3296   Because there is no discovery mechanism that can define this well-known namespace to a CIM client, it
3297   shall be one or more predefined namespaces. Therefore, to ensure interoperability, we recommend that
3298   pending future extensions of the WBEM specifications include discovery tools that define a namespace
3299   for these classes in a CIM server; these predefined namespaces should exist in either the root
3300   namespace or in the /root/CIMV2 namespace.

3301   A CIM server that supports capabilities reporting through these classes shall correctly report the current
3302   actual capabilities of the target CIM server and shall report on all capabilities defined. A CIM server is
3303   allowed to report "none" if the capability does not exist or "unknown" if the status of the capability is
3304   unknown at the time of the request for those properties where these choices exist in the properties
3305   definition. Because the CIM_ObjectManager object provides information on the target CIM server, only a
3306   single instance of this class may exist in a CIM server.

3307   The capabilities to be reported through the CIM_ObjectManagerCommunicationMechanism are as
3308   follows:

3309   •    CommunicationMechanism property, which defines the communication protocol for the
3310        CommunicationMechanism object. A compliant CIM server shall include the CIM-XML protocol
3311        for at least one ObjectManagerCommunicationMechanism instance.

3312   •    ProfilesSupported property, which defines the functional profiles supported as defined in clause
3313        5.3.4. All CIM servers shall support the basic-read functional group. All CIM clients may assume
3314        that any CIM server supports the basic-read functional group. The list of functional groups
3315        returned by a CIM server shall contain the basic-read group and shall not contain duplicates.
3316        CIM clients shall ignore duplicate entries in the functional-group list. If a functional group is
3317        included in the list, the CIM client shall assume that all other groups on which it depends
3318        (according to the rules defined in 5.3.4) are also supported. A CIM server should not explicitly
3319        include a functional group in the list whose presence may be inferred implicitly by a
3320        dependency. Support for a functional group does not imply that any method from that group will
3321        always succeed. Rather, the absence of the functional group from this list (whether explicit or
3322        implied) indicates to the CIM client that methods in that group will never succeed.

3323   •    MultipleOperationsSupported property, which defines whether the target CIM server supports
3324        multiple operation requests as defined in 5.3.2. `True` in this property indicates that the server
3325        can accept and process multiple operation requests. `False` indicates that the CIM server can
3326        accept only single operation requests.

3327   •    AuthenticationMechanismsSupported property, which defines the authentication mechanisms
3328        supported by the target CIM server as defined in 7.4.

3329   •    PulledEnumerationClosureOnExceedingServerLimits property, which indicates whether the CIM
3330        server supports closure of Pulled Enumeration sessions based upon exceeding server limits.

3331   •    PulledEnumerationContinuationOnErrorSupported property, which indicates whether the CIM
3332        server supports continuation on error for Pulled enumerations.

3333      •      PulledEnumerationMinimumOperationTimeout (PulledEnumerationMaximumOperationTimeout)
3334            property, which indicates the minimum (maximum) operation timeout allowed by the CIM server
3335            for Pulled enumerations.

3336 Compliant CIM servers may report additional capabilities for the CommunicationMechanism Functional
3337 Profiles, QueryLanguageSupported, and AuthenticationMechanismSupported by defining the "other"
3338 enumeration in the property and returning additional information in the associated "additional capabilities"
3339 property.

### 3340   7.5.2    Determining CIM Server Capabilities through the HTTP Options

3341 A CIM client may use the OPTIONS method to determine the CIM capabilities (if any) of the target server.
3342 A CIM server may support the OPTIONS method (for example, CIM servers supporting only HTTP/1.0
3343 would not support OPTIONS).

3344 To support the ability for a server to declare its CIM capabilities independently of HTTP, the DMTF
3345 intends to publish a CIM schema (in a separate document) describing such capabilities. In particular, this
3346 mechanism would allow servers that do not support the OPTIONS method to declare their capabilities to
3347 a client.

3348 If a CIM server supports the OPTIONS method, it should return the following headers in the response:

3349      •      CIM Extension Header CIMProtocolVersion, which provides a way for a client to discover the
3350            version of the CIM HTTP mapping supported by the CIM server.

3351      •      CIM Extension Header CIMSupportedFunctionalGroups, which provides a way for a client to
3352            discover the CIM operations supported by the CIM server.

3353      •      CIM Extension Header CIMSupportsMultipleOperations, which provides a way for the client to
3354            discover whether the CIM server can support Multiple Operation Requests.

3355 In addition, if the CIM server supports one or more query languages, it should return the following header
3356 in the response:

3357      •      CIM Extension Header CIMSupportedQueryLanguages, which allows the client to discover the
3358            query languages supported by the CIM server.

3359 In addition, if the CIM server runs in a fixed validation mode, it should return the following header in the
3360 response:

3361      •      CIM Extension Header CIMValidation, which allows the client to determine whether the CIM
3362            server is strictly validating or loosely validating.

3363 If the CIMProtocolVersion, CIMSupportedFunctionalGroups, CIMSupportsMultipleOperations,
3364 CIMValidation, or CIMSupportedQueryLanguages extension headers are included in the response, the
3365 CIM server shall declare them as optional extension headers using the "Opt" header defined in
3366 RFC 2774.

3367 The full format of the "Opt" header declaration for this specification is:

```
3368    Opt              = "Opt" ":" "http://www.dmtf.org/cim/mapping/http/v1.0"
3369                         ";" "ns" "=" header-prefix
3370    header-prefix    = 2*DIGIT
```

3371 This header-prefix should be generated at random on a per-HTTP message basis and should not
3372 necessarily be a specific number.

3373 EXAMPLE: The following is a fragment of a legitimate OPTIONS response from a CIM server:

```
3374    HTTP/1.1 200 OK
3375    Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=77
```

```
3376        77-CIMProtocolVersion: 1.0
3377        77-CIMSupportedFunctionalGroups: basic-read
3378        77-CIMBatch
3379        77-CIMSupportedQueryLanguages: wql
3380        ...
```

3381 7.5.2.1   CIMSupportedFunctionalGroups

3382 The CIMSupportedFunctionalGroups extension header should be returned by a <u>CIM server</u> in any
3383 OPTIONS response. It shall not be returned in any other scenario.

3384 This header is defined as follows:

```
3385        CIMSupportedFunctionalGroups = "CIMSupportedFunctionalGroups" ":" 1#functional-
3386        group
3387        functional-group = "basic-read" |
3388            "basic-write" |
3389            "schema-manipulation" |
3390            "instance-manipulation" |
3391            "qualifier-declaration" |
3392            "association-traversal" |
3393            "query-execution"
```

3394 The functional group definitions correspond directly to those listed in 5.4.3. All CIM servers shall support
3395 the basic-read functional group. All <u>CIM clients</u> may assume that any CIM server supports the basic-read
3396 functional group.

3397 The list of functional groups returned by a CIM server shall contain the basic-read group and shall not
3398 contain any duplicates. CIM clients shall ignore any duplicate entries in the functional-group list.

3399 If a functional group is included in the list, the CIM client shall assume that all other groups on which it
3400 depends (according to the rules defined in 5.4.3) are also supported. A CIM server should not explicitly
3401 include a functional group in the list if the presence of the group may be implied by a dependency.

3402 EXAMPLE: The following HTTP response message indicates that the CIM server supports instance-manipulation,
3403   association-traversal, basic-write, and basic-read.

```
3404        HTTP/1.1 200 OK
3405        Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=77
3406        77-CIMProtocolVersion: 1.0
3407        77-CIMSupportedFunctionalGroups: association-traversal, instance-manipulation
3408        ...
```

3409 Support for a functional group does *not* imply that any method from that group will always succeed.
3410 Rather, the absence (whether explicit or implied) of the functional group from this header is an indication
3411 to the CIM client that methods in that group will *never* succeed.

3412 7.5.2.2   CIMSupportsMultipleOperations

3413 The CIMSupportsMultipleOperations extension header shall be returned in an OPTIONS response by any
3414 <u>CIM server</u> that supports <u>Multiple Operation Requests</u>. It shall not be returned in any other circumstances.

3415 This header is defined as follows:

```
3416        CIMSupportsMultipleOperations  = "CIMSupportsMultipleOperations"
```

3417 The presence of this header indicates that the server can accept and process multiple operation requests.
3418 The absence of this header indicates that the server can only accept and process <u>Simple Operation</u>
3419 <u>Requests</u>.

3420 **7.5.2.3 CIMSupportedQueryLanguages**

3421 The CIMSupportedQueryLanguages extension header should be returned by a CIM server that supports
3422 at least one query language in any OPTIONS response. It shall not be returned in any other scenario.

3423 This header is defined as follows (`token` has the meaning conferred by RFC 2068, section 2.2):

```
3424     CIMSupportedQueryLanguages  = "CIMSupportedQueryLanguages" ":" 1#query-language
3425     query-language     = token
```

3426 The query-language value shall be treated as case-insensitive. It is anticipated that query languages will
3427 be submitted for approval to the DMTF, and each submission will define a value for this token to enable it
3428 to be specified in this header.

3429 **7.5.2.4 CIMValidation**

3430 The CIMValidation extension header may be returned by a CIM server to provide information about the
3431 level of validation of CIM Operation Request messages.

3432 This header is defined as follows:

```
3433     CIMValidation = "CIMValidation" ":" validation-level
3434     validation-level = "validating" |
3435        "loosely-validating"
```

3436 A validation-level of `validating` indicates that the CIM server always applies strict validation of each
3437 CIM operation request. A validation-level of `loosely-validating` indicates that the CIM server applies
3438 loose validation of each CIM operation request.

3439 In the absence of this header, a CIM client should assume that the CIM server operates in strict validation
3440 mode.

3441 **7.6 Other HTTP Methods**

3442 This specification does not in any way define or constrain the way a CIM client, CIM server, or
3443 CIM listener uses any HTTP method other than those explicitly cited.

3444 **7.7 Discovery and Addressing**

3445 The target URI of the CIM Operation Request is defined as the location of the CIM server. This
3446 specification does not constrain the format of this URI other than it should be a valid URI (RFC 2396) for
3447 describing an HTTP-addressable resource.

3448 An HTTP server that supports the CIM mapping defined in this specification, and which supports the
3449 OPTIONS method, should include the following CIM extension header in an OPTIONS response:

3450 • CIMOM

3451 This header is defined as follows:
```
3452     CIMOM       = "CIMOM" ":" (absoluteURI | relativeURI)
```

3453 The terms absoluteURI and relativeURI are taken from RFC 2068; they indicate the location of the CIM
3454 server for this HTTP server.

3455 If the CIMOM extension header is included in the response, the CIM server shall declare it an optional
3456 extension header as described in 7.5.

3457 A CIM client that needs to communicate with a CIM server on an HTTP server should try an OPTIONS
3458 request to that HTTP server. If the OPTIONS request fails or the response does not include the CIM-

3459   CIMOM extension header, the CIM client may assume that the value of CIM-CIMOM is the relative URI
3460   **cimom**.

3461   The DMTF recommends the use of the following well-known IP ports in compliant CIM servers. This is a
3462   recommendation and not a requirement. The DMTF has registered these port addresses with IANA, so
3463   they are for the exclusive use of the DMTF.

3464   • CIM-XML (HTTP)   5988/tcp

3465   • CIM-XML (HTTP)   5988/udp

3466   • CIM-XML (HTTPS)   5989/tcp

3467   • CIM-XML (HTTPS)   5989/udp

3468   Other discovery mechanisms are outside the scope of this version of the specification.

3469   EXAMPLE 1:

3470   This example shows an HTTP server located at http://www.dmtf.org/ issuing an OPTIONS response to an HTTP
3471   client to indicate that its CIM server is located at http://www.dmtf.org/access/cimom.

```
3472       HTTP/1.1 200 OK
3473       Opt: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=48
3474       48-CIMOM: /access/cimom
3475       ...
```

3476   EXAMPLE 2:

3477   If an HTTP server located at http://www.dmtf.org/ responds with a "501 Not Implemented" to an OPTIONS
3478   request from a CIM client, the CIM client may then try to contact the CIM server at http://www.dmtf.org/cimom.

3479   7.8   Internationalization Considerations

3480   This clause defines the capabilities of the CIM HTTP mapping with respect to IETF policy guidelines on
3481   character sets and languages (RFC 2277).

3482   In this specification, human-readable fields are contained within a response or request entity body. In all
3483   cases, a human-readable content is encoded using XML (which explicitly provides for character set
3484   tagging and encoding) and requires that XML processors read XML elements encoded, at minimum,
3485   using the UTF-8 (RFC 2279) encoding of the ISO 10646 multilingual plane.

3486   Properties that are not of type string or string array shall not be localized.

3487   Because keys are writeable only on instantiation, key values shall not be localized. See DSP0004 for
3488   details.

3489   XML examples in this specification demonstrate the use of the charset parameter of the Content-Type
3490   header, as defined in RFC 2616, as well as the XML attribute on the <?xml> processing instruction, which
3491   together provide charset identification information for MIME and XML processors. This specification
3492   mandates that conforming applications shall support at least the "UTF-8" charset encoding (RFC 2277) in
3493   the Content-Type header and shall support the "UTF-8" value for the XML `encoding` attribute.

3494   XML also provides a language tagging capability for specifying the language of the contents of a
3495   particular XML element, based on use of IANA registered language tags (RFC 1766) in combination with
3496   ISO 639-1, in the xml:lang attribute of an XML element to identify the language of its content and
3497   attributes. Section 3.10 of RFC 2616 defines how the two-character ISO 639-1 language code is used as
3498   the primary-tag. The language-tag shall be registered by IANA.

3499   DSP0201 and DSP0203 declare this attribute on any XML elements. Therefore, conforming applications
3500   should use this attribute when specifying the language in which a particular element is encoded for string
3501   and string array attributes and qualifiers. See the usage rules on this element, which are defined by the

3502 World Wide Web Consortium in *XML 1.0, second edition*. The attribute may be scoped by the instance or
3503 a class and should not be scoped by a property because instances or classes should be localized in one
3504 language.

3505 This specification defines several names of HTTP headers and their values. These names are
3506 constructed using standard encoding practices so that they always have an HTTP-safe ASCII
3507 representation. Because these headers are not usually visible to users, they do not need to support
3508 encoding in multiple character sets.

3509 DSP0201 and DSP0203 introduce several XML element names. Similarly, these names are not visible to
3510 an end user and do not need to support multiple character set encodings.

3511 The CIM model (DSP0004) defines the subset of the Unicode character set that can be used to name
3512 CIM elements (classes, instances, methods, properties, qualifiers, and method parameters). In general,
3513 these characters appear as the value of XML attributes or as element content and are not displayed to
3514 end users.

3515 Negotiation and notification of language settings is effected in this mapping using the standard Accept-
3516 Language and Content-Language headers defined in RFC 2068.

3517   **ANNEX A**
3518   (Informative)

3519

3520

3521   **Examples of Message Exchanges**

3522   This annex illustrates the protocol defined in this document with examples of valid HTTP
3523   request/response exchanges. The examples are for illustration purposes only and are not considered part
3524   of the specification.

3525   For clarity, additional white space is included in the examples, but such white space is not an intrinsic part
3526   of such XML documents.

A.1   Retrieval of a Single Class Definition

3528   The following HTTP request illustrates how a client requests the class CIM_VideoBIOSElement.

```
3529       M-POST /cimom HTTP/1.1
3530       HOST: http://www.myhost.com/
3531       Content-Type: application/xml; charset=utf-8
3532       Content-Length: xxxx
3533       Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3534       73-CIMOperation: MethodCall
3535       73-CIMMethod: GetClass
3536       73-CIMObject: root/cimv2

3537       <?xml version="1.0" encoding="utf-8" ?>
3538       <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3539           <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3540               <SIMPLEREQ>
3541                   <IMETHODCALL NAME="GetClass">
3542                    <LOCALNAMESPACEPATH>
3543                           <NAMESPACE NAME="root"/>
3544                           <NAMESPACE NAME="cimv2"/>
3545                    </LOCALNAMESPACEPATH>
3546                    <IPARAMVALUE NAME="ClassName"><CLASSNAME
3547       NAME="CIM_VideoBIOSElement"/></IPARAMVALUE>
3548                       <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
3549                   </IMETHODCALL>
3550               </SIMPLEREQ>
3551           </MESSAGE>
3552       </CIM>
```

3553   Following is an HTTP response to the preceding request indicating success of the requested operation.
3554   For clarity of exposition, the complete definition of the returned <CLASS> element is not shown.

```
3555       HTTP/1.1 200 OK
3556       Content-Type: application/xml; charset=utf-8
3557       Content-Length: xxxx
3558       Ext:
3559       Cache-Control: no-cache
3560       Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3561       73-CIMOperation: MethodResponse
```

```
3562        <?xml version="1.0" encoding="utf-8" ?>
3563        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3564           <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3565              <SIMPLERSP>
3566                 <IMETHODRESPONSE NAME="GetClass">
3567                  <IRETURNVALUE>
3568                     <CLASS NAME="CIM_VideoBIOSElement" SUPERCLASS="CIM_SoftwareElement">
3569                     ...
3570                     </CLASS>
3571                  </IRETURNVALUE>
3572                 </IMETHODRESPONSE>
3573              </SIMPLERSP>
3574           </MESSAGE>
3575        </CIM>
```

3576 ## A.2   Retrieval of a Single Instance Definition

3577 The following HTTP request illustrates how a client requests the instance MyClass.MyKey="S3".

```
3578        M-POST /cimom HTTP/1.1
3579        HOST: http://www.myhost.com/
3580        Content-Type: application/xml; charset=utf-8
3581        Content-Length: xxxx
3582        Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3583        73-CIMOperation: MethodCall
3584        73-CIMMethod: GetInstance
3585        73-CIMObject: root%2FmyNamespace

3586        <?xml version="1.0" encoding="utf-8" ?>
3587        <CIM CIMVERSION="2.0" DTDVERSION="1.1">
3588           <MESSAGE ID="87855" PROTOCOLVERSION="1.0">
3589              <SIMPLEREQ>
3590                 <IMETHODCALL NAME="GetInstance">
3591                  <LOCALNAMESPACEPATH>
3592                     <NAMESPACE NAME="root"/>
3593                     <NAMESPACE NAME="myNamespace"/>
3594                  </LOCALNAMESPACEPATH>
3595                  <IPARAMVALUE NAME="InstanceName">
3596                     <INSTANCENAME CLASSNAME="MyClass">
3597                        <KEYBINDING NAME="MyKey"><KEYVALUE>S3</KEYVALUE></KEYBINDING>
3598                     </INSTANCENAME>
3599                  </IPARAMVALUE>
3600                  <IPARAMVALUE NAME="LocalOnly"><VALUE>FALSE</VALUE></IPARAMVALUE>
3601                 </IMETHODCALL>
3602              </SIMPLEREQ>
3603           </MESSAGE>
3604        </CIM>
```

3605 Following is an HTTP response to the preceding request indicating an error because the specified
3606 instance is not found.

```
3607        HTTP/1.1 200 OK
3608        Content-Type: application/xml; charset=utf-8
3609        Content-Length: xxxx
3610        Ext:
```

```
3611    Cache-Control: no-cache
3612    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3613    73-CIMOperation: MethodResponse

3614    <?xml version="1.0" encoding="utf-8" ?>
3615    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3616        <MESSAGE ID="87885" PROTOCOLVERSION="1.0">
3617            <SIMPLERSP>
3618                <IMETHODRESPONSE NAME="GetInstance">
3619                 <ERROR CODE="6" DESCRIPTION="Instance of MyClass not found"/>
3620                </IMETHODRESPONSE>
3621            </SIMPLERSP>
3622        </MESSAGE>
3623    </CIM>
```

3624   A.3   Deletion of a Single Class Definition

3625   The following HTTP request illustrates how a client deletes the class CIM_VideoBIOSElement.

```
3626    M-POST /cimom HTTP/1.1
3627    HOST: http://www.myhost.com/
3628    Content-Type: application/xml; charset=utf-8
3629    Content-Length: xxxx
3630    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3631    73-CIMOperation: MethodCall
3632    73-CIMMethod: DeleteClass
3633    73-CIMObject: root/cimv2

3634    <?xml version="1.0" encoding="utf-8" ?>
3635    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3636        <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3637            <SIMPLEREQ>
3638                <IMETHODCALL NAME="DeleteClass">
3639                    <LOCALNAMESPACEPATH>
3640                     <NAMESPACE NAME="root"/>
3641                     <NAMESPACE NAME="cimv2"/>
3642                    </LOCALNAMESPACEPATH>
3643                    <IPARAMVALUE NAME="ClassName"><CLASSNAME
3644    NAME="CIM_VideoBIOSElement"/></IPARAMVALUE>
3645                </IMETHODCALL>
3646            </SIMPLEREQ>
3647        </MESSAGE>
3648    </CIM>
```

3649   Following is an HTTP response to the preceding request indicating failure of the preceding operation due
3650   to the inability to delete instances of the class.

```
3651    HTTP/1.1 200 OK
3652    Content-Type: application/xml; charset=utf-8
3653    Content-Length: xxxx
3654    Ext:
3655    Cache-Control: no-cache
3656    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3657    73-CIMOperation: MethodResponse

3658    <?xml version="1.0" encoding="utf-8" ?>
3659    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
```

```
3660          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3661              <SIMPLERSP>
3662                  <IMETHODRESPONSE NAME="DeleteClass">
3663                   <ERROR CODE="9" DESCRIPTION="Class has non-deletable instances"/>
3664                  </IMETHODRESPONSE>
3665              </SIMPLERSP>
3666          </MESSAGE>
3667      </CIM>
```

3668 **A.4   Deletion of a Single Instance Definition**

3669 The following HTTP request illustrates how a client deletes the instance MyClass.MyKey="S3".

```
3670      M-POST /cimom HTTP/1.1
3671      HOST: http://www.myhost.com/
3672      Content-Type: application/xml; charset=utf-8
3673      Content-Length: xxxx
3674      Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3675      73-CIMOperation: MethodCall
3676      73-CIMMethod: DeleteInstance
3677      73-CIMObject: root%2FmyNamespace

3678      <?xml version="1.0" encoding="utf-8" ?>
3679      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3680          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3681              <SIMPLEREQ>
3682                  <IMETHODCALL NAME="DeleteInstance">
3683                      <LOCALNAMESPACEPATH>
3684                          <NAMESPACE NAME="root"/>
3685                          <NAMESPACE NAME="myNamespace"/>
3686                      </LOCALNAMESPACEPATH>
3687                      <IPARAMVALUE NAME="InstanceName">
3688                          <INSTANCENAME CLASSNAME="MyClass">
3689                              <KEYBINDING NAME="MyKey">
3690                               <KEYVALUE>S3</KEYVALUE>
3691                              </KEYBINDING>
3692                          </INSTANCENAME>
3693                      </IPARAMVALUE>
3694                  </IMETHODCALL>
3695              </SIMPLEREQ>
3696          </MESSAGE>
3697      </CIM>
```

3698 Following is an HTTP response to the preceding request indicating success of the preceding operation.

```
3699      HTTP/1.1 200 OK
3700      Content-Type: application/xml; charset=utf-8
3701      Content-Length: xxxx
3702      Ext:
3703      Cache-Control: no-cache
3704      Man: http://www.dmtf.org/cim/operation ; ns=73
3705      73-CIMOperation: MethodResponse

3706      <?xml version="1.0" encoding="utf-8" ?>
3707      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3708          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
```

```
3709        <SIMPLERSP>
3710          <IMETHODRESPONSE NAME="DeleteInstance"/>
3711        </SIMPLERSP>
3712      </MESSAGE>
3713    </CIM>
```

A.5   Creation of a Single Class Definition

The following HTTP request illustrates how a client creates the class MySchema_VideoBIOSElement as a subclass of CIM_VideoBIOSElement. For clarity of exposition, most of the submitted <CLASS> element is omitted from the example.

```
M-POST /cimom HTTP/1.1
HOST: http://www.myhost.com/
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: CreateClass
73-CIMObject: root/cimv2

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
    <SIMPLEREQ>
      <IMETHODCALL NAME="CreateClass">
        <LOCALNAMESPACEPATH>
          <NAMESPACE NAME="root"/>
          <NAMESPACE NAME="cimv2"/>
        </LOCALNAMESPACEPATH>
        <IPARAMVALUE NAME="NewClass">
          <CLASS NAME="MySchema_VideoBIOSElement"
SUPERCLASS="CIM_VideoBIOSElement">
                 ...
          </CLASS>
        </IPARAMVALUE>
      </IMETHODCALL>
    </SIMPLEREQ>
  </MESSAGE>
</CIM>
```

Following is an HTTP response to the preceding request indicating success of the preceding operation.

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
Content-Length: xxxx
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodResponse

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
    <SIMPLERSP>
      <IMETHODRESPONSE NAME="CreateClass"/>
```

```
3758            </SIMPLERSP>
3759          </MESSAGE>
3760       </CIM>
```

3761 **A.6   Creation of a Single Instance Definition**

3762 The following HTTP request illustrates how a client creates an instance of the class
3763 MySchema_VideoBIOSElement. For clarity of exposition, most of the submitted <INSTANCE> element is
3764 omitted from the example.

```
3765      M-POST /cimom HTTP/1.1
3766      HOST: http://www.myhost.com/
3767      Content-Type: application/xml; charset=utf-8
3768      Content-Length: xxxx
3769      Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3770      73-CIMOperation: MethodCall
3771      73-CIMMethod: CreateInstance
3772      73-CIMObject: root/cimv2

3773      <?xml version="1.0" encoding="utf-8" ?>
3774      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3775         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3776            <SIMPLEREQ>
3777               <IMETHODCALL NAME="CreateInstance">
3778                  <LOCALNAMESPACEPATH>
3779                   <NAMESPACE NAME="root"/>
3780                   <NAMESPACE NAME="cimv2"/>
3781                  </LOCALNAMESPACEPATH>
3782                  <IPARAMVALUE NAME="NewInstance">
3783                   <INSTANCE CLASSNAME="CIM_VideoBIOSElement">
3784                         ...
3785                   </INSTANCE>
3786                  </IPARAMVALUE>
3787               </IMETHODCALL>
3788            </SIMPLEREQ>
3789         </MESSAGE>
3790      </CIM>
```

3791 Following is an HTTP response to the preceding request indicating the success of the preceding
3792 operation.

```
3793      HTTP/1.1 200 OK
3794      Content-Type: application/xml; charset=utf-8
3795      Content-Length: xxxx
3796      Ext:
3797      Cache-Control: no-cache
3798      Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3799      73-CIMOperation: MethodResponse

3800      <?xml version="1.0" encoding="utf-8" ?>
3801      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3802         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3803            <SIMPLERSP>
3804               <IMETHODRESPONSE NAME="CreateInstance">
3805                  <IRETURNVALUE>
3806                      <INSTANCENAME CLASSNAME="MySchema_VideoBIOSElement">
```

```
3807                          <KEYBINDING NAME="Name">
3808                           <KEYVALUE>S4</KEYVALUE>
3809                          </KEYBINDING>
3810                       </INSTANCENAME>
3811                   </IRETURNVALUE>
3812               </IRETURNVALUE>
3813           </SIMPLERSP>
3814       </MESSAGE>
3815     </CIM>
```

A.7   Enumeration of Class Names

The following HTTP request illustrates how a client enumerates the names of all subclasses of the class
CIM_SoftwareElement.

```
3819     M-POST /cimom HTTP/1.1
3820     HOST: http://www.myhost.com/
3821     Content-Type: application/xml; charset=utf-8
3822     Content-Length: xxxx
3823     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3824     73-CIMOperation: MethodCall
3825     73-CIMMethod: EnumerateClassNames
3826     73-CIMObject: root/cimv2

3827     <?xml version="1.0" encoding="utf-8" ?>
3828     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3829        <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3830           <SIMPLEREQ>
3831              <IMETHODCALL NAME="EnumerateClassNames">
3832                 <LOCALNAMESPACEPATH>
3833                  <NAMESPACE NAME="root"/>
3834                  <NAMESPACE NAME="cimv2"/>
3835                 </LOCALNAMESPACEPATH>
3836                 <IPARAMVALUE NAME="ClassName"><CLASSNAME
3837                     NAME="CIM_SoftwareElement"/></IPARAMVALUE>
3838                 <IPARAMVALUE
3839                     NAME="DeepInheritance"><VALUE>FALSE</VALUE></IPARAMVALUE>
3840              </IMETHODCALL>
3841           </SIMPLEREQ>
3842        </MESSAGE>
3843     </CIM>
```

Following is an HTTP response to the preceding request indicating the success of the preceding
operation and returning the names of the requested subclasses.

```
3846     HTTP/1.1 200 OK
3847     Content-Type: application/xml; charset=utf-8
3848     Content-Length: xxxx
3849     Ext:
3850     Cache-Control: no-cache
3851     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3852     73-CIMOperation: MethodResponse

3853     <?xml version="1.0" encoding="utf-8" ?>
3854     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3855        <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
```

```
3856              <SIMPLERSP>
3857                  <IMETHODRESPONSE NAME="EnumerateClassNames">
3858                      <IRETURNVALUE>
3859                       <CLASSNAME NAME="CIM_BIOSElement"/>
3860                       <CLASSNAME NAME="CIM_VideoBOISElement"/>
3861                      </IRETURNVALUE>
3862                  </IMETHODRESPONSE>
3863              </SIMPLERSP>
3864          </MESSAGE>
3865      </CIM>
```

3866   A.8   Enumeration of Instances

3867   The following HTTP request illustrates how a client enumerates all instances of the class
3868   CIM_LogicalDisk. For clarity of exposition, most of the returned instances are omitted from the example.

```
3869      M-POST /cimom HTTP/1.1
3870      HOST: http://www.myhost.com/
3871      Content-Type: application/xml; charset=utf-8
3872      Content-Length: xxxx
3873      Man: http://www.dmtf.org/cim/operation ; ns=73
3874      73-CIMOperation: MethodCall
3875      73-CIMMethod: EnumerateInstances
3876      73-CIMObject: root/cimv2

3877      <?xml version="1.0" encoding="utf-8" ?>
3878      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3879          <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3880              <SIMPLEREQ>
3881                  <IMETHODCALL NAME="EnumerateInstances">
3882                      <LOCALNAMESPACEPATH>
3883                       <NAMESPACE NAME="root"/>
3884                       <NAMESPACE NAME="cimv2"/>
3885                      </LOCALNAMESPACEPATH>
3886                      <IPARAMVALUE NAME="ClassName"><CLASSNAME
3887                          NAME="CIM_LogicalDisk"/></IPARAMVALUE>
3888                      <IPARAMVALUE NAME="LocalOnly"><VALUE>TRUE</VALUE></IPARAMVALUE>
3889                      <IPARAMVALUE NAME="DeepInheritance"><VALUE>TRUE</VALUE></IPARAMVALUE>
3890                  </IMETHODCALL>
3891              </SIMPLEREQ>
3892          </MESSAGE>
3893      </CIM>
```

3894   Following is an HTTP response to the preceding request indicating success of the preceding operation,
3895   returning the requested instances.

```
3896      HTTP/1.1 200 OK
3897      Content-Type: application/xml; charset=utf-8
3898      Content-Length: xxxx
3899      Ext:
3900      Cache-Control: no-cache
3901      Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3902      73-CIMOperation: MethodResponse

3903      <?xml version="1.0" encoding="utf-8" ?>
3904      <CIM CIMVERSION="2.0" DTDVERSION="2.0">
```

```
3905        <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3906          <SIMPLERSP>
3907            <IMETHODRESPONSE NAME="EnumerateInstances">
3908              <IRETURNVALUE>
3909                <VALUE.NAMEDINSTANCE>
3910                  <INSTANCENAME CLASSNAME="Erewhon_LogicalDisk">
3911                    ...
3912                  </INSTANCENAME>
3913                  <INSTANCE CLASSNAME="Erewhon_LogicalDisk">
3914                    ...
3915                  </INSTANCE>
3916                </VALUE.NAMEDINSTANCE>
3917                ...
3918                <VALUE.NAMEDINSTANCE>
3919                  <INSTANCENAME CLASSNAME="Foobar_LogicalDisk">
3920                    ...
3921                  </INSTANCENAME>
3922                  <INSTANCE CLASSNAME="Foobar_LogicalDisk">
3923                    ...
3924                  </INSTANCE>
3925                </VALUE.NAMEINSTANCE>
3926              </IRETURNVALUE>
3927            </IMETHODRESPONSE>
3928          </SIMPLERSP>
3929        </MESSAGE>
3930    </CIM>
```

A.9   Retrieval of a Single Property

3932 The following HTTP request illustrates how a client retrieves the FreeSpace property from the instance
3933 MyDisk.DeviceID="C:".

```
3934    M-POST /cimom HTTP/1.1
3935    HOST: http://www.myhost.com/
3936    Content-Type: application/xml; charset=utf-8
3937    Content-Length: xxxx
3938    Man: http://www.dmtf.org/cim/operation ; ns=73
3939    73-CIMOperation: MethodCall
3940    73-CIMMethod: GetProperty
3941    73-CIMObject: root%2FmyNamespace

3942    <?xml version="1.0" encoding="utf-8" ?>
3943    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3944      <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3945        <SIMPLEREQ>
3946          <IMETHODCALL NAME="GetProperty">
3947            <LOCALNAMESPACEPATH>
3948              <NAMESPACE NAME="root"/>
3949              <NAMESPACE NAME="myNamespace"/>
3950            </LOCALNAMESPACEPATH>
3951            <IPARAMVALUE NAME="InstanceName">
3952              <INSTANCENAME CLASSNAME="MyDisk">
3953                <KEYBINDING NAME="DeviceID"><KEYVALUE>C:</KEYVALUE></KEYBINDING>
3954              </INSTANCENAME>
```

```
3955              </IPARAMVALUE>
3956              <IPARAMVALUE
3957    NAME="PropertyName"><VALUE>FreeSpace</VALUE></IPARAMVALUE>
3958             </IMETHODCALL>
3959          </SIMPLEREQ>
3960        </MESSAGE>
3961    </CIM>
```

3962 Following is an HTTP response to the preceding request indicating success of the preceding operation,
3963 returning the requested value.

```
3964    HTTP/1.1 200 OK
3965    Content-Type: application/xml; charset=utf-8
3966    Content-Length: xxxx
3967    Ext:
3968    Cache-Control: no-cache
3969    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3970    73-CIMOperation: MethodResponse

3971    <?xml version="1.0" encoding="utf-8" ?>
3972    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3973        <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3974            <SIMPLERSP>
3975                <IMETHODRESPONSE NAME="GetProperty">
3976                    <IRETURNVALUE>
3977                     <VALUE>6752332</VALUE>
3978                    </IRETURNVALUE>
3979                </IMETHODRESPONSE>
3980            </SIMPLERSP>
3981        </MESSAGE>
3982    </CIM>
```

3983 A.10   Execution of an Extrinsic Method

3984 The following HTTP request illustrates how a client executes the SetPowerState method on the instance
3985 MyDisk.DeviceID="C:".

```
3986    M-POST /cimom HTTP/1.1
3987    HOST: http://www.myhost.com/
3988    Content-Type: application/xml; charset=utf-8
3989    Content-Length: xxxx
3990    Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
3991    73-CIMOperation: MethodCall
3992    73-CIMMethod: SetPowerState
3993    73-CIMObject: root%2FmyNamespace%3AMyDisk.Name%3D%22C%3A%22

3994    <?xml version="1.0" encoding="utf-8" ?>
3995    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
3996        <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
3997            <SIMPLEREQ>
3998                <METHODCALL NAME="SetPowerState">
3999                    <LOCALINSTANCEPATH>
4000                        <LOCALNAMESPACEPATH>
4001                         <NAMESPACE NAME="root"/>
4002                         <NAMESPACE NAME="myNamespace"/>
4003                        </LOCALNAMESPACEPATH>
```

```
4004                          <INSTANCENAME CLASSNAME="MyDisk">
4005                           <KEYBINDING NAME="Name"><KEYVALUE>C:</KEYVALUE></KEYBINDING>
4006                          </INSTANCENAME>
4007                      </LOCALINSTANCEPATH>
4008                      <PARAMVALUE NAME="PowerState"><VALUE>1</VALUE></PARAMVALUE>
4009                      <PARAMVALUE
4010     NAME="Time"><VALUE>00000001132312.000000:000</VALUE></PARAMVALUE>
4011                  </METHODCALL>
4012              </SIMPLEREQ>
4013          </MESSAGE>
4014      </CIM>
```

4015 Following is an HTTP response to the preceding request indicating the success of the preceding
4016 operation.

```
4017     HTTP/1.1 200 OK
4018     Content-Type: application/xml; charset=utf-8
4019     Content-Length: xxxx
4020     Ext:
4021     Cache-Control: no-cache
4022     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
4023     73-CIMOperation: MethodResponse

4024     <?xml version="1.0" encoding="utf-8" ?>
4025     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4026         <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
4027             <SIMPLERSP>
4028                 <METHODRESPONSE NAME="SetPowerState">
4029                     <RETURNVALUE>
4030                      <VALUE>0</VALUE>
4031                     </RETURNVALUE>
4032                 </METHODRESPONSE>
4033             </SIMPLERSP>
4034         </MESSAGE>
4035     </CIM>
```

4036 ## A.11 Indication Delivery Example

4037 The following HTTP request illustrates the format for sending an indication of type CIM_AlertIndication to
4038 a CIM listener.

```
4039     M-POST /cimlistener/browser HTTP/1.1
4040     HOST: http://www.acme.com/
4041     Content-Type: application/xml; charset=utf-8
4042     Content-Length: XXX
4043     Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=40
4044     40-CIMExport: MethodRequest
4045     40-CIMExportMethod: ExportIndication

4046     <?xml version="1.0" encoding="utf-8" ?>
4047     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4048         <MESSAGE ID="1007" PROTOCOLVERSION="1.0">
4049             <SIMPLEEXPREQ>
4050                 <EXPMETHODCALL NAME="ExportIndication">
4051                     <EXPPARAMVALUE NAME="NewIndication">
4052                         <INSTANCE CLASSNAME="CIM_AlertIndication" >
```

```
4053                             <PROPERTY NAME="Description" TYPE="string">
4054                                 <VALUE>Sample CIM_AlertIndication indication</VALUE>
4055                             </PROPERTY>
4056                             <PROPERTY NAME="AlertType" TYPE="uint16">
4057                                 <VALUE>1</VALUE>
4058                             </PROPERTY>
4059                               <PROPERTY NAME="PerceivedSeverity" TYPE="uint16">
4060                               <VALUE>3</VALUE>
4061                             </PROPERTY>
4062                               <PROPERTY NAME="ProbableCause" TYPE="uint16">
4063                               <VALUE>2</VALUE>
4064                             </PROPERTY>
4065                               <PROPERTY NAME="IndicationTime" TYPE="datetime">
4066                               <VALUE>20010515104354.000000:000</VALUE>
4067                             </PROPERTY>
4068                         </INSTANCE>
4069                     </EXPPARAMVALUE>
4070                 </EXPMETHODCALL>
4071             </SIMPLEEXPREQ>
4072         </MESSAGE>
4073     </CIM>
```

4074 Following is an HTTP response to the preceding request indicating a successful receipt by the CIM
4075 listener.

```
4076     HTTP/1.1 200 OK
4077     Content-Type: application/xml; charset=utf-8
4078     Content-Length: 267
4079     Ext:
4080     Cache-Control: no-cache
4081     Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=40
4082     40-CIMExport: MethodResponse

4083     <?xml version="1.0" encoding="utf-8" ?>
4084     <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4085         <MESSAGE ID="1007" PROTOCOLVERSION="1.0">
4086             <SIMPLEEXPRSP>
4087                 <EXPMETHODRESPONSE NAME="ExportIndication">
4088                     <IRETURNVALUE>
4089                     </IRETURNVALUE>
4090                 </EXPMETHODRESPONSE>
4091             </SIMPLEEXPRSP>
4092         </MESSAGE>
4093     </CIM>
```

4094   **A.12**   **Subscription Example**

4095 A CIM client application activates a subscription by creating an instance of the
4096 CIM_IndicationSubscription class, which defines an association between a CIM_IndicationFilter (a filter)
4097 instance and a CIM_IndicationHandler (a handler) instance. The CIM_IndicationFilter instance defines the
4098 filter criteria and data project list to describe the desired indication stream. The CIM_IndicationHandler
4099 instance defines the desired indication encoding, destination location, and protocol for delivering the
4100 indication stream.

4101   The following HTTP request illustrates how a client creates an instance of the class CIM_IndicationFilter.
4102   Note that the exact syntax of the WMI Query Language is still under review and is subject to change.

```
4103        Host: bryce
4104        Content-Type: application/xml; charset=utf-8
4105        Content-Length: XXXX
4106        Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=20
4107        20-CIMProtocolVersion: 1.0
4108        20-CIMOperation: MethodCall
4109        20-CIMMethod: CreateInstance
4110        20-CIMObject: root/cimv2

4111        <?xml version="1.0" encoding="utf-8"?>
4112        <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4113           <MESSAGE ID="53000" PROTOCOLVERSION="1.0">
4114              <SIMPLEREQ>
4115                 <IMETHODCALL NAME="CreateInstance">
4116                    <LOCALNAMESPACEPATH>
4117                       <NAMESPACE NAME="root"/>
4118                       <NAMESPACE NAME="cimv2"/>
4119                    </LOCALNAMESPACEPATH>
4120                    <IPARAMVALUE NAME="NewInstance">
4121                       <INSTANCE CLASSNAME="CIM_IndicationFilter">
4122                          <PROPERTY NAME="SystemCreationClassName" TYPE="string">
4123                             <VALUE>CIM_UnitaryComputerSystem</VALUE>
4124                          </PROPERTY>
4125                          <PROPERTY NAME="SystemName" TYPE="string">
4126                             <VALUE>server001.acme.com</VALUE>
4127                          </PROPERTY>
4128                          <PROPERTY NAME="CreationClassName" TYPE="string">
4129                             <VALUE>CIM_IndicationFilter</VALUE>
4130                          </PROPERTY>
4131                          <PROPERTY NAME="Name" TYPE="string">
4132                             <VALUE>ACMESubscription12345</VALUE>
4133                          </PROPERTY>
4134                          <PROPERTY NAME="SourceNamespace" TYPE="string">
4135                             <VALUE>root/cimv2</VALUE>
4136                          </PROPERTY>
4137                          <PROPERTY NAME="Query" TYPE="string">
4138                             <VALUE>
4139                             SELECT Description, AlertType, PerceivedSeverity,
4140                                ProbableCause, IndicationTime
4141                                FROM CIM_AlertIndication
4142                                WHERE PerceivedSeverity = 3
4143                             </VALUE>
4144                          </PROPERTY>
4145                          <PROPERTY NAME="QueryLanguage" TYPE="string">
4146                             <VALUE>WQL</VALUE>
4147                          </PROPERTY>
4148                       </INSTANCE>
4149                    </IPARAMVALUE>
4150                 </IMETHODCALL>
4151              </SIMPLEREQ>
4152           </MESSAGE>
```

```
4153        </CIM>
```

4154 Following is an HTTP response to the preceding request indicating success of the preceding operation.

```
4155        HTTP/1.1 200 OK
4156        Content-Type: application/xml; charset=utf-8
4157        Content-Length: XXX
4158        Ext:
4159        Cache-Control: no-cache
4160        Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=28
4161        28-CIMOperation: MethodResponse
4162
4163        <?xml version="1.0" encoding="utf-8" ?>
4164          <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4165            <MESSAGE ID="53000" PROTOCOLVERSION="1.0">
4166              <SIMPLERSP>
4167                <IMETHODRESPONSE NAME="CreateInstance">
4168                  <IRETURNVALUE>
4169                  <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
4170                      <KEYBINDING NAME="SystemCreationClassName">
4171                        <KEYVALUE VALUETYPE="string">
4172                            CIM_UnitaryComputerSystem
4173                        </KEYVALUE>
4174                      </KEYBINDING>
4175                      <KEYBINDING NAME="SystemName">
4176                        <KEYVALUE VALUETYPE="string">
4177                            server001.acme.com
4178                        </KEYVALUE>
4179                      </KEYBINDING>
4180                      <KEYBINDING NAME="CreationClassName">
4181                        <KEYVALUE VALUETYPE="string">
4182                            CIM_IndicationFilter
4183                        </KEYVALUE>
4184                      </KEYBINDING>
4185                      <KEYBINDING NAME="Name">
4186                        <KEYVALUE VALUETYPE="string">
4187                            ACMESubscription12345
4188                        </KEYVALUE>
4189                    </KEYBINDING>
4190                    </INSTANCENAME>
4191                  </IRETURNVALUE>
4192                </IMETHODRESPONSE>
4193              </SIMPLERSP>
4194            </MESSAGE>
4195          </CIM>
```

4196 The following HTTP request illustrates how a client creates an instance of the class
4197 CIM_IndicationHandlerCIMXML.

```
4198        M-POST /cimom HTTP/1.1
4199        Host: bryce
4200        Content-Type: application/xml; charset=utf-8
4201        Content-Length: XXX
4202        Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=20
4203        20-CIMProtocolVersion: 1.0
```

```
4204       20-CIMOperation: MethodCall
4205       20-CIMMethod: CreateInstance
4206       20-CIMObject: root/cimv2
4207       <?xml version="1.0" encoding="utf-8"?>
4208       <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4209       <MESSAGE ID="54000" PROTOCOLVERSION="1.0">
4210             <SIMPLEREQ>
4211                 <IMETHODCALL NAME="CreateInstance">
4212                     <LOCALNAMESPACEPATH>
4213                         <NAMESPACE NAME="root"/>
4214                         <NAMESPACE NAME="cimv2"/>
4215                     </LOCALNAMESPACEPATH>
4216                     <IPARAMVALUE NAME="NewInstance">
4217                         <INSTANCE CLASSNAME="CIM_IndicationHandlerCIMXML">
4218                             <PROPERTY NAME="SystemCreationClassName" TYPE="string">
4219                                 <VALUE>CIM_UnitaryComputerSystem</VALUE>
4220                             </PROPERTY>
4221                             <PROPERTY NAME="SystemName" TYPE="string">
4222                                 <VALUE>server001.acme.com</VALUE>
4223                                 </PROPERTY>
4224                             <PROPERTY NAME="CreationClassName" TYPE="string">
4225                                 <VALUE>CIM_IndicationHandlerCIMXML</VALUE>
4226                             </PROPERTY>
4227                             <PROPERTY NAME="Name" TYPE="string">
4228                                 <VALUE>ACMESubscription12345</VALUE>
4229                             </PROPERTY>
4230                             <PROPERTY NAME="Owner" TYPE="string">
4231                                 <VALUE>ACMEAlertMonitoringConsole</VALUE>
4232                             </PROPERTY>
4233                             <PROPERTY NAME="Destination" TYPE="string">
4234                                 <VALUE>HTTP://www.acme.com/cimlistener/browser</VALUE>
4235                             </PROPERTY>
4236                         </INSTANCE>
4237                     </IPARAMVALUE>
4238                 </IMETHODCALL>
4239             </SIMPLEREQ>
4240         </MESSAGE>
4241     </CIM>
```

4242   Following is an HTTP response to the preceding request indicating the success of the preceding
4243   operation.

```
4244       HTTP/1.1 200 OK
4245       Content-Type: application/xml; charset=utf-8
4246       Content-Length: XXX
4247       Ext:
4248       Cache-Control: no-cache
4249       Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=27
4250       27-CIMOperation: MethodResponse
4251       <?xml version="1.0" encoding="utf-8" ?>
4252         <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4253             <MESSAGE ID="54000" PROTOCOLVERSION="1.0">
4254                 <SIMPLERSP>
```

```
4255                    <IMETHODRESPONSE NAME="CreateInstance">
4256                        <IRETURNVALUE>
4257                            <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
4258                                <KEYBINDING NAME="SystemCreationClassName">
4259                                    <KEYVALUE VALUETYPE="string">
4260                                        CIM_UnitaryComputerSystem
4261                                    </KEYVALUE>
4262                                </KEYBINDING>
4263                                <KEYBINDING NAME="SystemName">
4264                                    <KEYVALUE VALUETYPE="string">
4265                                        server001.acme.com
4266                                    </KEYVALUE>
4267                                </KEYBINDING>
4268                                <KEYBINDING NAME="CreationClassName">
4269                                    <KEYVALUE VALUETYPE="string">
4270                                        CIM_IndicationHandlerCIMXML
4271                                    </KEYVALUE>
4272                                </KEYBINDING>
4273                                <KEYBINDING NAME="Name">
4274                                    <KEYVALUE VALUETYPE="string">
4275                                        ACMESubscription12345
4276                                    </KEYVALUE>
4277                                </KEYBINDING>
4278                            </INSTANCENAME>
4279                        </IRETURNVALUE>
4280                    </IMETHODRESPONSE>
4281                </SIMPLERSP>
4282            </MESSAGE>
4283        </CIM>
```

4284 The following HTTP request illustrates how a client creates an instance of the class
4285 CIM_IndicationSubscription.

```
4286    M-POST /cimom HTTP/1.1
4287    Host: bryce
4288    Content-Type: application/xml; charset=utf-8
4289    Content-Length: XXXX
4290    Man: http://www.dmtf.org/cim/mapping/http/v1.0;ns=55
4291    55-CIMProtocolVersion: 1.0
4292    55-CIMOperation: MethodCall
4293    55-CIMMethod: CreateInstance
4294    55-CIMObject: root/cimv2

4295    <?xml version="1.0" encoding="utf-8"?>
4296    <CIM CIMVERSION="2.0" DTDVERSION="2.0">
4297        <MESSAGE ID="55000" PROTOCOLVERSION="1.0">
4298            <SIMPLEREQ>
4299                <IMETHODCALL NAME="CreateInstance">
4300                    <LOCALNAMESPACEPATH>
4301                        <NAMESPACE NAME="root"/>
4302                        <NAMESPACE NAME="cimv2"/>
4303                    </LOCALNAMESPACEPATH>
4304                    <IPARAMVALUE NAME="NewInstance">
4305                        <INSTANCE CLASSNAME="CIM_IndicationSubscription">
```

```
4306                         <PROPERTY.REFERENCE NAME="Filter"
4307                            REFERENCECLASS="CIM_IndicationFilter">
4308                            <VALUE.REFERENCE>
4309                                <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
4310                                    <KEYBINDING NAME="SystemCreationClassName">
4311                                        <KEYVALUE VALUETYPE="string">
4312                                         CIM_UnitaryComputerSystem
4313                                        </KEYVALUE>
4314                                    </KEYBINDING>
4315                                    <KEYBINDING NAME="SystemName">
4316                                        <KEYVALUE VALUETYPE="string">
4317                                         server001.acme.com
4318                                        </KEYVALUE>
4319                                    </KEYBINDING>
4320                                    <KEYBINDING NAME="CreationClassName">
4321                                        <KEYVALUE VALUETYPE="string">
4322                                         CIM_IndicationFilter
4323                                        </KEYVALUE>
4324                                    </KEYBINDING>
4325                                    <KEYBINDING NAME="Name">
4326                                        <KEYVALUE VALUETYPE="string">
4327                                            ACMESubscription12345
4328                                        </KEYVALUE>
4329                                    </KEYBINDING>
4330                                </INSTANCENAME>
4331                            </VALUE.REFERENCE>
4332                            </PROPERTY.REFERENCE>
4333                            <PROPERTY.REFERENCE NAME="Handler"
4334          REFERENCECLASS="CIM_IndicationHandler">
4335                                <VALUE.REFERENCE>
4336                                <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
4337                                    <KEYBINDING NAME="SystemCreationClassName">
4338                                        <KEYVALUE VALUETYPE="string">
4339                                            CIM_UnitaryComputerSystem
4340                                        </KEYVALUE>
4341                                    </KEYBINDING>
4342                                    <KEYBINDING NAME="SystemName">
4343                                        <KEYVALUE VALUETYPE="string">
4344                                            server001.acme.com
4345                                        </KEYVALUE>
4346                                    </KEYBINDING>
4347                                    <KEYBINDING NAME="CreationClassName">
4348                                        <KEYVALUE VALUETYPE="string">
4349                                            CIM_IndicationHandlerCIMXML
4350                                        </KEYVALUE>
4351                                    </KEYBINDING>
4352                                    <KEYBINDING NAME="Name">
4353                                        <KEYVALUE VALUETYPE="string">
4354                                            ACMESubscription12345
4355                                        </KEYVALUE>
4356                                    </KEYBINDING>
4357                                </INSTANCENAME>
4358                            </VALUE.REFERENCE>
```

```
4359                    </PROPERTY.REFERENCE>
4360                 </INSTANCE>
4361              </IPARAMVALUE>
4362           </IMETHODCALL>
4363        </SIMPLEREQ>
4364     </MESSAGE>
4365  </CIM>
```

Following is an HTTP response to the preceding request indicating the success of the preceding
operation.

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
Content-Length: XXXX
Ext:
Cache-Control: no-cache
Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=75
75-CIMOperation: MethodResponse

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
   <MESSAGE ID="55000" PROTOCOLVERSION="1.0">
      <SIMPLERSP>
         <IMETHODRESPONSE NAME="CreateInstance">
            <IRETURNVALUE>
               <INSTANCENAME CLASSNAME="CIM_IndicationSubscription">
                  <KEYBINDING NAME="Filter">
                     <VALUE.REFERENCE>
                        <INSTANCENAME CLASSNAME="CIM_IndicationFilter">
                           <KEYBINDING NAME="SystemCreationClassName">
                              <KEYVALUE VALUETYPE="string">
                                 CIM_UnitaryComputerSystem
                              </KEYVALUE>
                           </KEYBINDING>
                           <KEYBINDING NAME="SystemName">
                              <KEYVALUE VALUETYPE="string">
                                 server001.acme.com
                              </KEYVALUE>
                           </KEYBINDING>
                           <KEYBINDING NAME="CreationClassName">
                              <KEYVALUE VALUETYPE="string">
                                 CIM_IndicationFilter
                              </KEYVALUE>
                           </KEYBINDING>
                           <KEYBINDING NAME="Name">
                              <KEYVALUE VALUETYPE="string">
                                 ACMESubscription12345
                              </KEYVALUE>
                           </KEYBINDING>
                        </INSTANCENAME>
                     </VALUE.REFERENCE>
                  </KEYBINDING>
                  <KEYBINDING NAME="Handler">
                     <VALUE.REFERENCE>
```

```
4410                            <INSTANCENAME CLASSNAME="CIM_IndicationHandlerCIMXML">
4411                                <KEYBINDING NAME="SystemCreationClassName">
4412                                    <KEYVALUE VALUETYPE="string">
4413                                        CIM_UnitaryComputerSystem
4414                                    </KEYVALUE>
4415                                </KEYBINDING>
4416                                <KEYBINDING NAME="SystemName">
4417                                    <KEYVALUE VALUETYPE="string">
4418                                     server001.acme.com
4419                                    </KEYVALUE>
4420                                </KEYBINDING>
4421                                <KEYBINDING NAME="CreationClassName">
4422                                    <KEYVALUE VALUETYPE="string">
4423                                        CIM_IndicationHandlerCIMXML
4424                                    </KEYVALUE>
4425                                </KEYBINDING>
4426                                <KEYBINDING NAME="Name">
4427                                    <KEYVALUE VALUETYPE="string">
4428                                        ACMESubscription12345
4429                                    </KEYVALUE>
4430                                </KEYBINDING>
4431                            </INSTANCENAME>
4432                        </VALUE.REFERENCE>
4433                    </KEYBINDING>
4434                </INSTANCENAME>
4435            </IRETURNVALUE>
4436        </IMETHODRESPONSE>
4437      </SIMPLERSP>
4438    </MESSAGE>
4439  </CIM>
```

4440 A.13 Multiple Operations Example

4441 The following HTTP request illustrates how a client performs multiple operations. This example batches a
4442 GetClass, an EnumerateInstanceNames, and an EnumerateInstance operation on
4443 CIM_ObjectManagerAdapter.

```
4444    POST /CIMOM1 HTTP/1.1
4445    Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
4446    Content-Length: XXX
4447    Host: localhost:5988
4448    CIMOperation: MethodCall
4449    CIMProtocolVersion: 1.0
4450    Content-Type: application/xml; charset=utf-8
4451    CIMBatch: CIMBatch
4452    <?xml version="1.0" encoding="UTF-8"?>
4453    <CIM DTDVERSION="2.0" CIMVERSION="2.0">
4454       <MESSAGE ID="2004:2:5:1:1:11:41:1" PROTOCOLVERSION="1.0">
4455          <MULTIREQ>
4456             <SIMPLEREQ>
4457                <IMETHODCALL NAME="GetClass">
4458                   <LOCALNAMESPACEPATH>
4459                      <NAMESPACE NAME="interop" />
4460                   </LOCALNAMESPACEPATH>
```

```
4461                    <IPARAMVALUE NAME="ClassName">
4462                        <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
4463                    </IPARAMVALUE>
4464                    <IPARAMVALUE NAME="LocalOnly">
4465                        <VALUE>FALSE</VALUE>
4466                    </IPARAMVALUE>
4467                    <IPARAMVALUE NAME="IncludeClassOrigin">
4468                        <VALUE>TRUE</VALUE>
4469                    </IPARAMVALUE>
4470                </IMETHODCALL>
4471            </SIMPLEREQ>
4472            <SIMPLEREQ>
4473                <IMETHODCALL NAME="Associators">
4474                    <LOCALNAMESPACEPATH>
4475                        <NAMESPACE NAME="interop" />
4476                    </LOCALNAMESPACEPATH>
4477                    <IPARAMVALUE NAME="ObjectName">
4478                        <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
4479                    </IPARAMVALUE>
4480                    <IPARAMVALUE NAME="IncludeQualifiers">
4481                        <VALUE>TRUE</VALUE>
4482                    </IPARAMVALUE>
4483                    <IPARAMVALUE NAME="IncludeClassOrigin">
4484                        <VALUE>TRUE</VALUE>
4485                    </IPARAMVALUE>
4486                </IMETHODCALL>
4487            </SIMPLEREQ>
4488            <SIMPLEREQ>
4489                <IMETHODCALL NAME="EnumerateInstanceNames">
4490                    <LOCALNAMESPACEPATH>
4491                        <NAMESPACE NAME="interop" />
4492                    </LOCALNAMESPACEPATH>
4493                    <IPARAMVALUE NAME="ClassName">
4494                        <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
4495                    </IPARAMVALUE>
4496                </IMETHODCALL>
4497            </SIMPLEREQ>
4498            <SIMPLEREQ>
4499                <IMETHODCALL NAME="EnumerateInstances">
4500                    <LOCALNAMESPACEPATH>
4501                        <NAMESPACE NAME="interop" />
4502                    </LOCALNAMESPACEPATH>
4503                    <IPARAMVALUE NAME="ClassName">
4504                        <CLASSNAME NAME="CIM_ObjectManagerAdapter" />
4505                    </IPARAMVALUE>
4506                    <IPARAMVALUE NAME="LocalOnly">
4507                        <VALUE>FALSE</VALUE>
4508                    </IPARAMVALUE>
4509                </IMETHODCALL>
4510            </SIMPLEREQ>
4511        </MULTIREQ>
4512    </MESSAGE>
4513 </CIM>
```

4514   Following is the HTTP response to the preceding request indicating the success of the preceding
4515   operation.

```
4516        HTTP/1.1 200 OK
4517        CIMOperation: MethodResponse
4518        Content-Length: XXX
4519        <?xml version="1.0" encoding="UTF-8"?>
4520        <CIM DTDVERSION="2.0" CIMVERSION="2.0">
4521          <MESSAGE ID="2004:2:5:1:1:11:41:1" PROTOCOLVERSION="1.0">
4522            <MULTIRSP>
4523              <SIMPLERSP>
4524                <IMETHODRESPONSE NAME="GetClass">
4525                  <IRETURNVALUE>
4526                    <CLASS SUPERCLASS="CIM_WBEMService"
4527                      NAME="CIM_ObjectManagerAdapter">
4528                      ...
4529                    </CLASS>
4530                  </IRETURNVALUE>
4531                </IMETHODRESPONSE>
4532              </SIMPLERSP>
4533              <SIMPLERSP>
4534                <IMETHODRESPONSE NAME="Associators">
4535                  <IRETURNVALUE>
4536                    <VALUE.OBJECTWITHPATH>
4537                      ...
4538                    </VALUE.OBJECTWITHPATH>
4539                    <VALUE.OBJECTWITHPATH>
4540                      ...
4541                    </VALUE.OBJECTWITHPATH>
4542                    ...
4543                  </IRETURNVALUE>
4544                </IMETHODRESPONSE>
4545              </SIMPLERSP>
4546              <SIMPLERSP>
4547                <IMETHODRESPONSE NAME="EnumerateInstanceNames">
4548                  <IRETURNVALUE>
4549                    <INSTANCENAME CLASSNAME="WBEMSolutions_ObjectManagerAdapter">
4550                      ...
4551                    </INSTANCENAME>
4552                    <INSTANCENAME CLASSNAME="WBEMSolutions_ObjectManagerAdapter">
4553                      ...
4554                    </INSTANCENAME>
4555                    ...
4556                  </IRETURNVALUE>
4557                </IMETHODRESPONSE>
4558              </SIMPLERSP>
4559              <SIMPLERSP>
4560                <IMETHODRESPONSE NAME="EnumerateInstances">
4561                  <IRETURNVALUE>
4562                    <VALUE.NAMEDINSTANCE>
4563                      ...
4564                    </VALUE.NAMEDINSTANCE>
4565                    <VALUE.NAMEDINSTANCE>
```

```
4566                              ...
4567                        </VALUE.NAMEDINSTANCE>
4568                              ...
4569                      </IRETURNVALUE>
4570                   </IMETHODRESPONSE>
4571                </SIMPLERSP>
4572             </MULTIRSP>
4573          </MESSAGE>
4574       </CIM>
```

4575   **ANNEX B**

4576   (informative)

4577

4578

4579   **LocalOnly Parameter Discussion**

4580   This annex discusses the issues associated with the 1.1 definition of the LocalOnly parameter for the
4581   GetInstance and EnumerateInstances operations.

4582   **B.1   Explanation of the Deprecated 1.1 Interpretation**

4583   In April 2002, two DMTF Change Requests (CRs), CR809 (EnumerateInstances) and CR815
4584   (GetInstance), were approved and incorporated into the 1.1 version of this specification to clarify the
4585   interpretation of the LocalOnly flag for the GetInstance and EnumerateInstances operations. With these
4586   CRs, the definition of the LocalOnly flag for these operations was modified to align with the interpretation
4587   of this flag for the GetClass and EnumerateClasses operations. This change was incorrect, resulted in
4588   reduced functionality, and introduced several backward compatibility issues.

4589   To clarify the difference between the 1.0 Interpretation and the 1.1 Interpretation (CR815), consider the
4590   following example:

```
4591        class A {
4592              [Key]
4593           string name;
4594           uint32 counter = 3;
4595        };

4596        class B : A {
4597           uint32 moreData = 4;
4598        };

4599        instance of A {
4600           name = "Roger";
4601        };

4602        instance of B {
4603           name = "Karl";
4604           counter = 3;
4605           moreData = 5;
4606        };

4607        instance of B {
4608           name = "Denise";
4609           counter = 5;
4610        };
```

4611 Assuming PropertyList = NULL and LocalOnly = TRUE, Table B-1 shows the properties returned by a
4612 GetInstance operation.

4613 **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

4614 The properties returned using the 1.0 interpretation are consistent with the properties specified in the
4615 MOF instance definitions, and the properties returned using the 1.1 Interpretation are consistent with the
4616 properties defined in the class definitions.

B.2 Risks of Using the 1.1 Interpretation

4617

4618 The risks of using the 1.1 interpretation are as follows:

4619 1) Within the DMTF, promoting a property from a class to one of its superclasses is defined as a
4620 backward-compatible change that can be made in a minor revision of the CIM schema. With the 1.1
4621 interpretation, promoting a property to a superclass can cause backward-incompatible changes.

4622 Suppose, for example, version 1.0 of the schema includes the following definitions:

```
4623    class A {
4624          [Key]
4625       string name;
4626       uint32 counter = 3;
4627    };

4628    class B : A {
4629       uint32 moreData = 4;
4630    };
```

4631 Now suppose that the schema is modified in version 1.1 to promote the property moreData from
4632 class B to class A.

```
4633    class A {
4634          [Key]
4635       string name;
4636       uint32 counter = 3;
4637       uint32 moreData = 4;
4638    };

4639    class B : A {

4640    };
```

4641 Using these examples, Table B-2 shows the properties returned by a call to GetInstance with
4642 PropertyList = NULL and LocalOnly = TRUE. With the 1.1 Interpretation, this schema change would
4643 affect the list of properties returned. When dealing with a CIM server that complies with the 1.1
4644 interpretation, applications must be designed to treat "promoting properties" as a backward-
4645 compatible change.

4646 **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

4647 2) The 1.1 Interpretation encourages application developers to use multiple operations to retrieve the
4648 properties of an instance. That is, a commonly-stated use model for the 1.1 interpretation is to
4649 selectively traverse subclasses getting additional properties of an instance. This practice significantly
4650 increases the risk that a client will construct an inconsistent instance. With both Interpretations,
4651 applications should be designed to ensure that dependent properties are retrieved together.

4652 **B.3 Techniques for Differentiating between the 1.0 Interpretation and 1.1**
4653 **Interpretation**

4654 For concrete classes, CIM servers that comply with the 1.0 Interpretation return the value of all KEY
4655 properties not explicitly excluded by the PropertyList parameter. CIM servers that comply with the 1.1
4656 interpretation return only the value of KEY properties explicitly defined in the class. Applications can use
4657 this difference to detect which interpretation is supported by a CIM server.

4658
4659

ANNEX C
(informative)

4660
4661
4662

Change Log

Version	Date	Description
Version 1.0	Jun 2, 1999	DMTF Final Standard
Version 1.1	Jan 06, 2003	DMTF Final Standard
Version 1.2	Jan 09, 2007	DMTF Final Standard
Version 1.3.0	Oct 15, 2008	DMTF Final Standard
Version 1.3.1	July 29, 2009	DMTF Standard Release

4663