



**Document Number: DSP0219**

**Date: 2009-08-11**

**Version: 1.0.0**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13

# **UML Profile for CIM**

14 **Document Type: Specification**

15 **Document Status: DMTF Standard**

16 **Document Language: E**

17

18 Copyright Notice

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

20 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems  
21 management and interoperability. Members and non-members may reproduce DMTF specifications and  
22 documents, provided that correct attribution is given. As DMTF specifications may be revised from time to  
23 time, the particular version and release date should always be noted.

24 Implementation of certain elements of this standard or proposed standard may be subject to third party  
25 patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations  
26 to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose,  
27 or identify any or all such third party patent right, owners or claimants, nor for any incomplete or  
28 inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to  
29 any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize,  
30 disclose, or identify any such third party patent rights, or for such party's reliance on the standard or  
31 incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any  
32 party implementing such standard, whether such implementation is foreseeable or not, nor to any patent  
33 owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is  
34 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  
35 implementing the standard from any and all claims of infringement by a patent owner for such  
36 implementations.

37 For information about patents held by third-parties which have notified the DMTF that, in their opinion,  
38 such patent may relate to or impact implementations of DMTF standards, visit  
39 <http://www.dmtf.org/about/policies/disclosures.php>.

40

# Contents

42	Foreword .....	6
43	Introduction .....	7
44	1 Scope .....	9
45	2 Normative References.....	9
46	2.1 Approved References .....	9
47	2.2 Other References.....	9
48	3 Terms and Definitions .....	10
49	3.1 General Rules .....	10
50	3.2 Definitions Related to CIM Qualifier Values .....	10
51	3.3 Other Terms and Definitions .....	10
52	3.4 Usage of ABNF .....	11
53	4 General Definition of a UML Profile.....	11
54	5 Definition of the UML Profile for CIM.....	12
55	5.1 UML Elements Used .....	12
56	5.2 Mapping of CIM Elements .....	14
57	5.3 Mapping to UML Packages.....	15
58	5.4 Mapping of CIM Classes.....	17
59	5.5 Mapping of CIM Indications .....	19
60	5.6 Mapping of CIM Associations .....	19
61	5.7 Mapping of CIM Properties .....	20
62	5.8 Mapping of CIM References .....	21
63	5.9 Mapping of CIM Methods.....	22
64	5.10 Mapping of CIM Parameters and CIM Return Values .....	23
65	5.11 Mapping of CIM Datatypes .....	25
66	5.12 Mapping of CIM Instances .....	26
67	5.13 Mapping of CIM Qualifiers .....	27
68	5.13.1 Overview .....	27
69	5.13.2 Mapping of Qualifier Types.....	29
70	5.13.3 Generic Mapping of Qualifier Values .....	33
71	5.13.4 Direct Mapping of Qualifier Values .....	34
72	5.13.5 Definition of Marker Stereotypes .....	47
73	5.13.6 Typical Use Cases for Qualifier Inheritance Mapping .....	49
74	5.14 Mapping MOF File Information .....	50
75	5.14.1 Stereotypes Used for Mapping MOF File Information .....	51
76	5.15 Stereotypes for the CIM Meta Elements.....	57
77	5.16 Other UML Metaclasses Used in the Mapping .....	59
78	5.16.1 UML <i>LiteralBoolean</i> Metaclass .....	59
79	5.16.2 UML <i>LiteralString</i> Metaclass.....	59
80	5.16.3 UML <i>LiteralInteger</i> Metaclass .....	60
81	5.16.4 UML <i>LiteralUnlimitedNatural</i> Metaclass.....	61
82	5.16.5 UML <i>LiteralNull</i> Metaclass .....	61
83	5.16.6 UML <i>Generalization</i> Metaclass.....	62
84	5.16.7 UML <i>DataType</i> Metaclass .....	63
85	5.17 Constraints.....	63
86	5.17.1 Additional OCL Functions .....	64
87	5.17.2 OCL Constraints on Classes, Associations, Indications and their Instances .....	67
88	5.17.3 OCL Constraints on Properties and References .....	70
89	5.17.4 OCL Constraints on Methods and Parameters.....	75
90	5.17.5 OCL Constraints on Qualifiers .....	77
91	5.17.6 OCL Constraints on Datatypes .....	103
92	5.17.7 OCL Constraints for Values of Directly Mapped Qualifiers .....	105
93	5.17.8 Other OCL Constraints .....	111

94	5.17.9 Other Constraints.....	116
95	5.17.10 Constraints Covered by UML.....	116
96	5.18 Extension Points .....	116
97	5.18.1 Attributes and Associations in UML Metaclasses Used for the Mapping .....	116
98	5.18.2 Additional Metamodel Constraints .....	117
99	5.18.3 Additional Stereotypes .....	117
100	5.19 Compatibility Considerations .....	117
101	5.20 Limitations.....	118
102	5.21 Definition of UML Profiles and Type Libraries .....	119
103	5.21.1 Profile "CIM".....	119
104	5.21.2 Profile "CIMQualifierType".....	119
105	5.21.3 Type Library "CIMDatatypes" .....	119
106	Bibliography .....	126
107		
108	<b>Figures</b>	
109	Figure 1 – Example for Package Mapping.....	16
110	Figure 2 – Example for Replacement of UML Class During Conversion of CIM MOF to UML .....	17
111	Figure 3 – Inheritance of Qualifier Related Stereotypes.....	121
112	Figure 4 – Contents of UML Profile "CIMQualifierType" .....	122
113	Figure 5 – UML Metaclasses Representing a CIM Class (high level) .....	123
114	Figure 6 – UML Metaclass Instances for a Simple CIM Class.....	124
115	Figure 7 – CIM MOF Definition of a Simple CIM Class .....	124
116		
117	<b>Tables</b>	
118	Table 1 – Overview of CIM Element Mapping .....	14
119	Table 2 – UML <i>Class</i> Metaclass .....	18
120	Table 3 – UML <i>AssociationClass</i> Metaclass.....	19
121	Table 4 – UML <i>Property</i> Metaclass Used to Map CIM Properties .....	20
122	Table 5 – UML <i>Property</i> Metaclass Used to Map CIM References .....	22
123	Table 6 – UML <i>Operation</i> Metaclass Used to Map CIM Methods .....	22
124	Table 7 – UML <i>Parameter</i> Metaclass Used to Map CIM Parameters and CIM Return Values.....	23
125	Table 8 – Mapping of CIM Datatypes .....	25
126	Table 9 – UML <i>InstanceSpecification</i> Metaclass Used for Modeled CIM Instances .....	26
127	Table 10 – UML <i>Slot</i> Metaclass Used for Property Values of Modeled CIM Instances.....	27
128	Table 11 – UML <i>Stereotype</i> Metaclass Used as Qualifier Type Stereotype .....	29
129	Table 12 – UML <i>Property</i> Metaclass Used for Definitional Aspects of CIM Qualifier Value Within Qualifier	
130	Type Stereotype.....	30
131	Table 13 – Properties of Stereotype <i>CIM_QualifierType</i> .....	32
132	Table 14 – Literals of Enumeration <i>CIM_QualifierInheritanceRule</i> .....	32
133	Table 15 – Qualifier Scope Stereotypes .....	33
134	Table 16 – Applied Meta Element Stereotypes Used for CIM Qualifier Values.....	34
135	Table 17 – Mapping of <i>Association/Aggregation/Composition/Aggregate</i> Qualifiers .....	35
136	Table 18 – Valid Combinations for <i>Association/Aggregation/Composition</i> Qualifiers .....	35
137	Table 19 – Mapping of <i>ArrayType</i> Qualifier .....	36
138	Table 20 – Literals of Enumeration <i>CIM_Qualifier_ArrayType_Enum</i> .....	36
139	Table 21 – UML <i>Comment</i> Metaclass Used in Multiple UML Metaclasses .....	37
140	Table 22 – <i>CIM_ClassReferenceType</i> Stereotype Used for <i>Property</i> and <i>Parameter</i> Metaclasses.....	37
141	Table 23 – Properties of Stereotype <i>CIM_ClassReferenceType</i> .....	38

142 Table 24 – Literals of Enumeration *CIM\_ClassReferenceType\_Enum* ..... 38

143 Table 25 – Mapping of *In/Out* Qualifiers ..... 39

144 Table 26 – Mapping of OCL Constraint Related Qualifiers..... 40

145 Table 27 – Usage of UML *Constraint* Metaclass for OCL Related Constraint Qualifiers ..... 41

146 Table 28 – Usage of UML *OpaqueExpression* Metaclass Used for OCL Related Constraint Qualifiers... 42

147 Table 29 – UML *Enumeration* Metaclass Used for *Values* and *ValueMap* Qualifiers ..... 44

148 Table 30 – *CIM\_Enumeration* Stereotype Used for *Enumeration* Metaclass Used for *Values* and

149 *ValueMap* Qualifiers..... 45

150 Table 31 – Properties of Stereotype *CIM\_Enumeration*..... 46

151 Table 32 – UML *EnumerationLiteral* Metaclass Used for Pair of *Values* and *ValueMap* Qualifier Values 46

152 Table 33 – Mapping of *Write* Qualifier ..... 47

153 Table 34 – UML *Stereotype* Metaclass Used as Marker Stereotype..... 48

154 Table 35 – UML *Image* Metaclass Used by Marker Stereotype ..... 48

155 Table 36 – Mapping of CIM MOF File Related Information ..... 50

156 Table 37 – Stereotype *CIM\_ClassLocale* ..... 51

157 Table 38 – Property *ClassLocale* Owned by Stereotype *CIM\_ClassLocale*..... 52

158 Table 39 – Stereotype *CIM\_InstanceLocale*..... 52

159 Table 40 – Property *InstanceLocale* Owned by Stereotype *CIM\_InstanceLocale* ..... 53

160 Table 41 – Stereotype *CIM\_NamespacePath* ..... 53

161 Table 42 – Property *NamespacePath* Owned by Stereotype *CIM\_NamespacePath* ..... 54

162 Table 43 – Stereotype *CIM\_MofFileInfo* ..... 54

163 Table 44 – Property *FileName* Owned by Stereotype *CIM\_MofFileInfo*..... 55

164 Table 45 – Property *DirectoryPath* Owned by Stereotype *CIM\_MofFileInfo* ..... 55

165 Table 46 – Property *HeaderLines* Owned by Stereotype *CIM\_MofFileInfo*..... 56

166 Table 47 – Other Definitions for Stereotypes Used for Mapping MOF File Information ..... 56

167 Table 48 – Other Definitions for Properties of Stereotypes Used for Mapping MOF File Information..... 57

168 Table 49 – Meta Element Stereotypes..... 58

169 Table 50 – Definition of Meta Element Stereotypes ..... 58

170 Table 51 – UML *LiteralBoolean* Metaclass ..... 59

171 Table 52 – UML *LiteralString* Metaclass ..... 60

172 Table 53 – UML *LiteralInteger* Metaclass ..... 60

173 Table 54 – UML *LiteralUnlimitedNatural* Metaclass..... 61

174 Table 55 – UML *LiteralNull* Metaclass ..... 62

175 Table 56 – UML *Generalization* Metaclass ..... 62

176 Table 57 – UML *Data Type* Metaclass..... 63

177

178

## Foreword

179 DSP0219, *UML Profile for CIM*, was prepared by the DMTF Architecture Working Group in collaboration  
180 with the Object Management Group (OMG) under the alliance between DMTF and OMG.

### 181 Acknowledgments

182 DMTF acknowledges the following DMTF members for their contributions to this document:

- 183 • George Ericson – EMC
- 184 • Christoph Fehling – IBM
- 185 • Steve Jerman – Cisco
- 186 • Andreas Maier – IBM
- 187 • Kirk Wilson – CA, Inc.

188 DMTF acknowledges the following Object Management Group (OMG) members for their contributions to  
189 this document:

- 190 • Dusko Mistic – IBM
- 191 • Pete Rivett – Adaptive
- 192 • Bran Selic – IBM

193 Participants from the DMTF Architecture Working Group:

- 194 • George Ericson – EMC
- 195 • Christoph Fehling – IBM
- 196 • Steve Jerman – Cisco
- 197 • Andreas Maier – IBM
- 198 • Jeff Piazza – Hewlett-Packard Company
- 199 • Kirk Wilson – CA, Inc.

200 Participants from the Object Management Group's Architecture Board:

- 201 • Sridhar Iyengar – IBM
- 202 • Jishnu Mukerji – Hewlett-Packard Company
- 203 • Pete Rivett – Adaptive
- 204 • Bran Selic – IBM

205

206

## Introduction

207 The Unified Modeling Language (UML) from the Object Management Group (OMG) allows users to  
208 specify, visualize and document software systems. It includes twelve diagram types to allow various  
209 aspects of a system's design to be modeled. These diagrams are:

- 210 • **Structural Diagrams** include the Class Diagram, Object Diagram, Component Diagram, and  
211 Deployment Diagram.
- 212 • **Behavior Diagrams** include the Use Case Diagram (used by some methodologies during  
213 requirements gathering); Sequence Diagram, Activity Diagram, Collaboration Diagram, and  
214 Statechart Diagram.
- 215 • **Model Management Diagrams** include Packages, Subsystems, and Models.

216 The CIM metamodel defined in the *CIM Infrastructure Specification* ([DSP0004](#)) is very similar to the UML  
217 metamodel. The DMTF has used the CIM metamodel to define the CIM Schema, which is a management  
218 schema that forms an ontology for computer and systems management. The CIM Schema defines a rich  
219 and detailed set of classes that establish a common framework for the description of the managed  
220 environment. The classes include methods as well as properties and so enable both active management  
221 and instrumentation. The CIM Schema is extensible so that vendors can define their own extensions by  
222 subclassing. Note that the CIM metamodel could also be used to define CIM models that are not part of  
223 the CIM Schema.

224 The purpose of this document is to formalize the relationship between CIM and UML.

225 The OMG has defined a four-layer metamodel architecture that is defined in the [OMG MOF Core](#)  
226 [Specification](#). These four metamodel layers are called M0 to M3. Within this architecture, UML is defined  
227 as a metamodel at the M2 metamodel level. The CIM metamodel could be defined as another M2  
228 metamodel; however, this would not meet the goal of allowing the use of standard UML tools. Instead,  
229 this document utilizes the defined UML extension points to map the CIM metamodel to a UML profile. A  
230 UML profile is a lightweight extension mechanism for the UML metamodel. This allows using standard  
231 UML modeling tools to develop any CIM models and the CIM schema.

232 One of the big advantages of using UML tools is that they provide a wealth of capabilities beyond just  
233 representing the CIM model. For example, they can be used to describe state diagrams, use cases or  
234 interactions between management applications and managed elements.

235 NOTE: The specific profile defined in this document is called the "UML Profile for CIM". It is a UML profile, not to be  
236 confused with DMTF management profiles. As a side note: The term "CIM profile" is ambiguous, is not defined by the  
237 DMTF, and should not be used for either kind of profile.





238

# UML Profile for CIM

## 239 1 Scope

240 This document defines a UML profile which expresses the DMTF Common Information Model (CIM) using  
241 the OMG Unified Modeling Language (UML).

242 The specified mapping allows automated conversion between the DMTF defined CIM MOF file format or  
243 other equivalent representations of a CIM model and a UML model. This conversion can be done in both  
244 directions without loss of information, supporting round-trip engineering.

245 The mapping also allows creation of an XML based representation of the UML Profile for CIM, for usage  
246 by UML tools. The [UML Superstructure Specification](#) and the [OMG MOF/XML Mapping Specification](#)  
247 together define an XML based file format for UML profiles.

248 This document is based on the CIM metamodel defined in the [CIM Infrastructure Specification,  
249 Version 2.5](#). This document supports any CIM schema versions based upon that CIM metamodel.

250 This document uses the UML metamodel defined in the [UML Superstructure Specification, Version 2.1.1](#).

## 251 2 Normative References

252 This document depends on the following other documents. For references to specific versions, only the  
253 version cited applies. For references to documents without specific versions, the latest version of the  
254 referenced document (including any amendments) applies.

### 255 2.1 Approved References

256 DMTF DSP0004, *CIM Infrastructure Specification 2.5*,  
257 [http://www.dmtf.org/standards/published\\_documents/DSP0004\\_2.5.pdf](http://www.dmtf.org/standards/published_documents/DSP0004_2.5.pdf)

258 *OMG UML Superstructure Specification, Version 2.1.1*,  
259 <http://www.omg.org/cgi-bin/doc?formal/07-02-05>

260 *OMG UML Infrastructure Specification, Version 2.1.1*,  
261 <http://www.omg.org/cgi-bin/doc?formal/07-02-06>

262 *OMG UML OCL Specification, Version 2.0*,  
263 <http://www.omg.org/cgi-bin/doc?formal/06-05-01>

264 *754-1985 IEEE Standard for Binary Floating-Point Arithmetic*,  
265 [http://shop.ieee.org/ieeestore/Product.aspx?product\\_no=SH10116](http://shop.ieee.org/ieeestore/Product.aspx?product_no=SH10116)

266 *IETF RFC5234, Augmented BNF for Syntax Specifications: ABNF* January 2008,  
267 <http://www.faqs.org/rfcs/rfc5234.html>

### 268 2.2 Other References

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

## 271 3 Terms and Definitions

### 272 3.1 General Rules

273 The key phrases and words *shall*, *shall not*, *should*, *should not*, *may*, *need not*, *can*, *cannot* in this  
274 document are to be interpreted as described in [Annex H](#) of the ISO rules for the structure and drafting of  
275 international standards.

276 Any text in this document is normative, unless noted otherwise. Any paragraphs starting with "Note ..."  
277 and any examples are non-normative.

### 278 3.2 Definitions Related to CIM Qualifier Values

#### 279 3.1

##### 280 **default value**

281 The default value for a qualifier as defined in the qualifier declaration (in CIM MOF or in UML).

#### 282 3.2

##### 283 **inherited value**

284 Defined only for qualifiers with flavor *ToSubclass*. The inherited value is the value in the next superclass  
285 (towards the top of the hierarchy) that has this qualifier defined. If there is no such superclass (i.e., the  
286 class is a top class), then it is the default value.

287 Note: As usual in CIM, properties and methods with the same name in a class hierarchy have an inheritance  
288 relationship only if they are connected via the *Override* qualifier.

#### 289 3.3

##### 290 **effective value**

291 The value a qualifier effectively has at a particular level in the class hierarchy, regardless of whether or  
292 not it has a value defined at that level.

293 The effective value is determined as follows:

- 294 • If a value for a qualifier is defined at the particular hierarchy level, the effective value is the  
295 value defined there.
- 296 • Otherwise, for qualifiers with flavor *ToSubclass* the effective value is the inherited value, and for  
297 qualifiers with flavor *Restricted* the effective value is the default value.

#### 298 3.4

##### 299 **defined value**

300 If a qualifier value is specified (or set) for a CIM element, it is said to be *defined* on the CIM element.

## 301 3.3 Other Terms and Definitions

### 302 3.3.1

#### 303 **CIM namespace**

304 A CIM namespace, as defined in the [CIM Infrastructure Specification](#).

### 305 3.3.2

#### 306 **covered property**

307 A property in a CIM class that has the same name as a property in one of its subclasses, without being  
308 overridden. This is an undesired but possible condition that can arise, for example if an extension schema  
309 has added a property to an extension subclass, and later, the base CIM schema adds a property with the  
310 same name to a CIM-defined base class. CIM instances then have occurrences of all these properties.

311 **3.3.3**312 **inheritance chain**

313 At the class level, an inheritance chain is the ordered set of a CIM class and all of its base classes. Since  
314 in CIM there is only single inheritance, the inheritance chain is a linear set of classes.

315 At the property and method level, an inheritance chain is the ordered set of overridden properties and  
316 methods, respectively.

317 **3.3.4**318 **package path**

319 Ordered set of packages in a package tree from a reference point (usually the top of the tree) to a  
320 particular package, along the containing packages. This is very similar to a directory path in a hierarchical  
321 file system. UML defines syntax for package paths by concatenating the package names, separated by  
322 two colons. Example: "CIM::Device::Network"

323 **3.4 Usage of ABNF**

324 This document uses [Augmented BNF](#) with the following exception:

- 325 • Rules separated by a bar ( | ) represent choices. (Instead of using a slash ( / ) as defined in  
326 ABNF).

327 ABNF defines that any items in rules be concatenated without inserting any implicit whitespace  
328 characters between them, so any intended whitespace characters need to be specified explicitly in the  
329 ABNF.

330 ABNF defines that any literal strings and rule names be treated case-insensitively.

331 **4 General Definition of a UML Profile**

332 A UML profile is an extension mechanism that defines how the UML metamodel is tailored for different  
333 purposes. This includes the ability to tailor the UML metamodel for different platforms (such as J2EE or  
334 .NET) or domains (such as real-time, business process, or resource modeling). The UML profile  
335 mechanism is not a first-class extension mechanism in the way that it would allow for modifying the UML  
336 metamodel. Rather, it is a lightweight extension mechanism that defines specific extension points for the  
337 UML metamodel.

338 A UML profile is usually defined by means of a UML profile specification (like this document). UML tools  
339 are expected to create any artifacts that are needed in order to represent and implement the UML profile.  
340 Since starting with UML 2.0, a UML profile is represented by a UML metaclass *Profile*, UML profiles can  
341 now be exchanged between UML tools via XML.

342 The *Profiles* package in the [UML Superstructure Specification](#) defines the extension mechanisms  
343 supported by UML profiles:

- 344 • Stereotypes – Allows to extend UML metaclasses with additional attributes (called "Tagged  
345 Values" in earlier UML versions)
- 346 • Additional metamodel constraints – allows to define additional constraints between UML  
347 metaclasses (For example, that only single inheritance is supported). These constraints can be  
348 expressed in natural language or in a formal language such as UML's Object Constraint  
349 Language (OCL).

350 Note: The *Classes* package already defines an extension mechanism, namely the Model Library mechanism which is  
351 typically used in UML profile specifications in order to define standard datatypes for the tailored UML metamodel.

352 This UML profile specification tailors the UML metamodel to represent the CIM metamodel.

## 353 5 Definition of the UML Profile for CIM

354 This clause normatively defines the UML profile for CIM.

355 The transformation of CIM MOF to a UML user model is fully defined by this clause, with the limitations  
356 defined in 5.20. This transformation does not lose any information.

357 The transformation of a UML user model that has the CIM profile for UML applied to CIM MOF is fully  
358 defined by this clause, with the limitations defined in 5.20. This transformation does not lose any  
359 information if the UML user model is a *pure CIM model*.

360 A UML user model that has the CIM profile for UML applied is a *pure CIM model* if it

- 361 • contains only instances of the UML metaclasses listed in 5.1 and
- 362 • satisfies all constraints defined in 5.17 and
- 363 • does not utilize any of the extension points defined in 5.18.

364 For example, the UML profile for CIM maps CIM association classes to UML association classes and  
365 does not use straight UML associations. Therefore, if a UML user model contains straight UML  
366 associations, it is not a pure CIM model. A tool or an automated transformation (such as XSLT on the XML  
367 or [OMG MOF QVT](#)) could detect the usage of a straight UML association and could convert it into an  
368 association class.

369 As another example, using UML diagrams also causes the UML user model to no longer be a pure CIM  
370 model. Note that it makes perfect sense to amend a UML user model with diagrams; however, these  
371 diagrams are not representable in CIM MOF.

372 A UML user model that has the CIM profile for UML applied is a *valid CIM model* if it satisfies all  
373 constraints defined in 5.17.

### 374 5.1 UML Elements Used

375 The UML profile for CIM uses the following UML metaclasses, and implicitly all their super-metaclasses.  
376 Metaclasses used as attributes or as associated metaclasses of those used directly, are also shown.

377 From the [UML Superstructure Specification](#):

- 378 • From the UML::Classes::Kernel package:
  - 379 – AggregationKind
  - 380 – Class
  - 381 – Comment
  - 382 – Constraint
  - 383 – ElementImport
  - 384 – Generalization
  - 385 – InstanceSpecification
  - 386 – LiteralBoolean
  - 387 – LiteralInteger
  - 388 – LiteralNull
  - 389 – LiteralString
  - 390 – LiteralUnlimitedNatural

- 391           – Operation
- 392           – Package
- 393           – PackageImport
- 394           – Parameter
- 395           – ParameterDirectionKind
- 396           – DataType
- 397           – Property (merged with AssociationClasses::Property)
- 398           – Slot
- 399           – VisibilityKind
- 400           • From the UML::Classes::AssociationClasses package:
  - 401           – AssociationClass
  - 402           – Property (merged with Kernel::Property)
- 403           • From the UML::Profiles package:
  - 404           – Image
  - 405           – Profile
  - 406           – Stereotype

407 From the [UML Infrastructure Specification](#):

- 408           • From the InfrastructureLibrary::Core::Abstractions package:
  - 409           – VisibilityKind
- 410           • From the InfrastructureLibrary::Core::Basic package:
- 411           • From the InfrastructureLibrary::Core::Constructs package:
  - 412           – Property

413 The UML compliance level required by this document is UML L2 plus the Classes::AssociationClasses  
414 package.

415 Sometimes, UML metaclasses are defined in multiple UML packages. (For example, the UML *Class*  
416 metaclass has a superclass *Classifier* which is defined in both the *Classes::Kernel* and the  
417 *Classes::Dependencies* package.) Such metaclasses usually have a *package merge* relationship that  
418 defines that one of the metaclasses merges the other. The package merge concept is defined in 7.3.40  
419 (PackageMerge (from Kernel)) of the [UML Superstructure Specification](#). For the UML *Property* metaclass,  
420 this means that even though there are two different packages defining a UML *Property* metaclass  
421 (*Kernel::Property* and *AssociationClasses::Property*) there is only one variant of the UML *Property*  
422 metaclass which is the "merge" of these two.

423 In this document, any reference to a UML metaclass means the merged variant as defined in the [UML](#)  
424 [Superstructure Specification](#).

425 **5.2 Mapping of CIM Elements**

426 Table 1 gives a non-normative overview of the mapping.

427 **Table 1 – Overview of CIM Element Mapping**

CIM Element	UML Construct
Class	<i>Class</i> metaclass, as defined in 5.4.
Indication	<i>Class</i> metaclass, as defined in 5.5.
Association	<i>AssociationClass</i> metaclass, as defined in 5.6.
Property	<i>Property</i> metaclass, as defined in 5.7.
Reference	<i>Property</i> metaclass, as defined in 5.8.
Method	<i>Operation</i> metaclass, as defined in 5.9.
Parameter	<i>Parameter</i> metaclass, as defined in 5.10.
Schema	<i>CIM_Schema</i> stereotype, as defined in 5.3, and in addition the schema name stays a prefix on class names.
Namespace	The CIM Namespace in a CIM Server is not mapped since it is a runtime entity. The <code>#pragma namespace</code> directive in CIM MOF files is mapped to the <i>CIM_NamespacePath</i> stereotype, as defined in 5.14.
Instance	<i>InstanceSpecification</i> metaclass, as defined in 5.12.
Datatype	<i>Data Type</i> metaclass, as defined in 5.11.
Qualifier	<i>Stereotype</i> metaclass and native UML constructs, as defined in 5.13.
UmlPackagePath Qualifier	<i>Package</i> hierarchy, as defined in 5.3.
OCL Constraint Qualifiers	<i>Constraint</i> metaclass, as defined in 5.13.4.
Values and ValueMap Qualifiers	<i>Enumeration</i> and <i>EnumerationLiteral</i> metaclasses, as defined in 5.13.4.16.
CIM MOF files	Several stereotypes, as defined in 5.14.
Trigger	not mapped, as explained in 5.20.

428 The following subclauses normatively define the mapping for each CIM element.

429 The tables used in these subclauses list all "non-derived" attributes and associations of the UML  
 430 metaclasses as defined in the [UML Superstructure Specification](#). UML attributes and associations that  
 431 are marked as "derived" in the [UML Superstructure Specification](#) can be calculated from other entities  
 432 and therefore are not listed in these tables.

433 Future versions of the [UML Superstructure Specification](#) may define additional UML attributes or  
 434 associations. If this is done in a downward compatible way, this version of this document can be used to  
 435 map that future version without change.

436 In the following subclauses, associations between UML metaclasses are only described in terms of those  
 437 of their association ends that are owned by the associated metaclasses. The associations themselves, as  
 438 well as any association ends owned by them are not described. This follows the way these associations  
 439 are described in the [UML Superstructure Specification](#).

440 NOTE: Associations between the UML metaclasses are not to be confused with UML associations (i.e., the  
 441 *AssociationClass* metaclass), as they link UML metaclasses (for example a *Property* to a *Class*), whereas UML  
 442 associations link instances of the UML *Class* metaclass (i.e., "classes").

443 For UML attributes and associations that are mapped to CIM elements, the mapping is mandatory unless  
 444 noted otherwise. For some of the attributes and associations that are not mapped to CIM elements, a  
 445 *compliance value* is defined. If the attribute or association does not have the compliance value, the UML

446 user model is not a valid CIM model. Tools may implement validation against the compliance value in  
447 order to expose error or warning conditions when validating the UML user model or when exporting the  
448 UML user model to CIM MOF.

449 The tables used in these subclauses list the multiplicities of UML attributes and associations. In some  
450 cases, they are restrictions of the multiplicities defined in the UML metaclasses. The default multiplicity  
451 used in these tables is "[1]". A multiplicity of "[0]" for UML associations means that there are no  
452 associated elements. A multiplicity of "[0]" for UML attributes means that there are no values.

### 453 **5.3 Mapping to UML Packages**

454 NOTE: In UML, any element is contained in a package. When mapping CIM elements to UML, it can be assumed  
455 without restricting the general applicability of this document that a target package is defined, which acts as the root of  
456 a package tree containing the CIM elements.

457 The UML package containing a CIM class (or indication or association) shall be determined by the  
458 *UmlPackagePath* qualifier of that class, as follows: The UML *Class* to which a CIM class is mapped shall  
459 be (directly) owned by a UML *Package* whose (relative) package path under the above mentioned target  
460 package is:

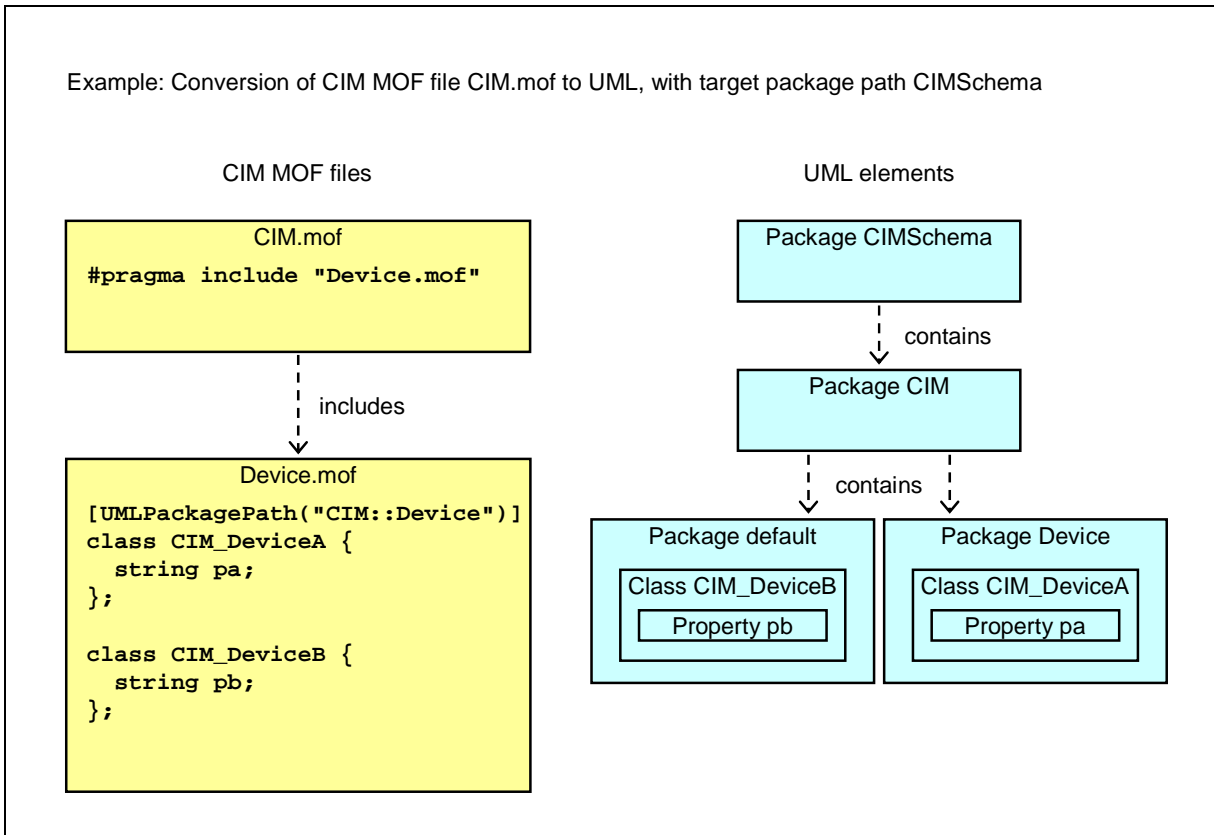
- 461 • the value of its *UmlPackagePath* qualifier if that value is not NULL, or
- 462 • otherwise, the schema name of the CIM class, followed by " : : default".

463 The first package in this package path shall have the stereotype *CIM\_Schema* applied. [DSP0004](#)  
464 mandates that the first package in a package path of a class is the schema name of the class.

465 Any package owned directly or indirectly by a package that has the stereotype *CIM\_Schema* applied,  
466 shall have the stereotype *CIM\_Package* applied. The *CIM\_Package* and *CIM\_Schema* stereotypes are  
467 defined in 5.15.

468 NOTE FOR IMPLEMENTATIONS: If the conversion of CIM MOF to UML involves replacing an existing UML class  
469 with the CIM class being converted, then replacement of the UML class should be done in a minimally invasive way,  
470 for instance by emptying it of all owned elements (i.e., *ownedAttribute*, *ownedOperation*, *ownedRule*,  
471 *ownedComment*), and recreating these elements as defined in the CIM class being converted. This is better than  
472 deleting and recreating the entire UML class because it allows retaining any references to the class in the underlying  
473 UML storage format, which is advantageous for supporting roundtrip engineering through a UML/CIM-MOF cycle.

474 Figure 1 shows an example for the package mapping.



475

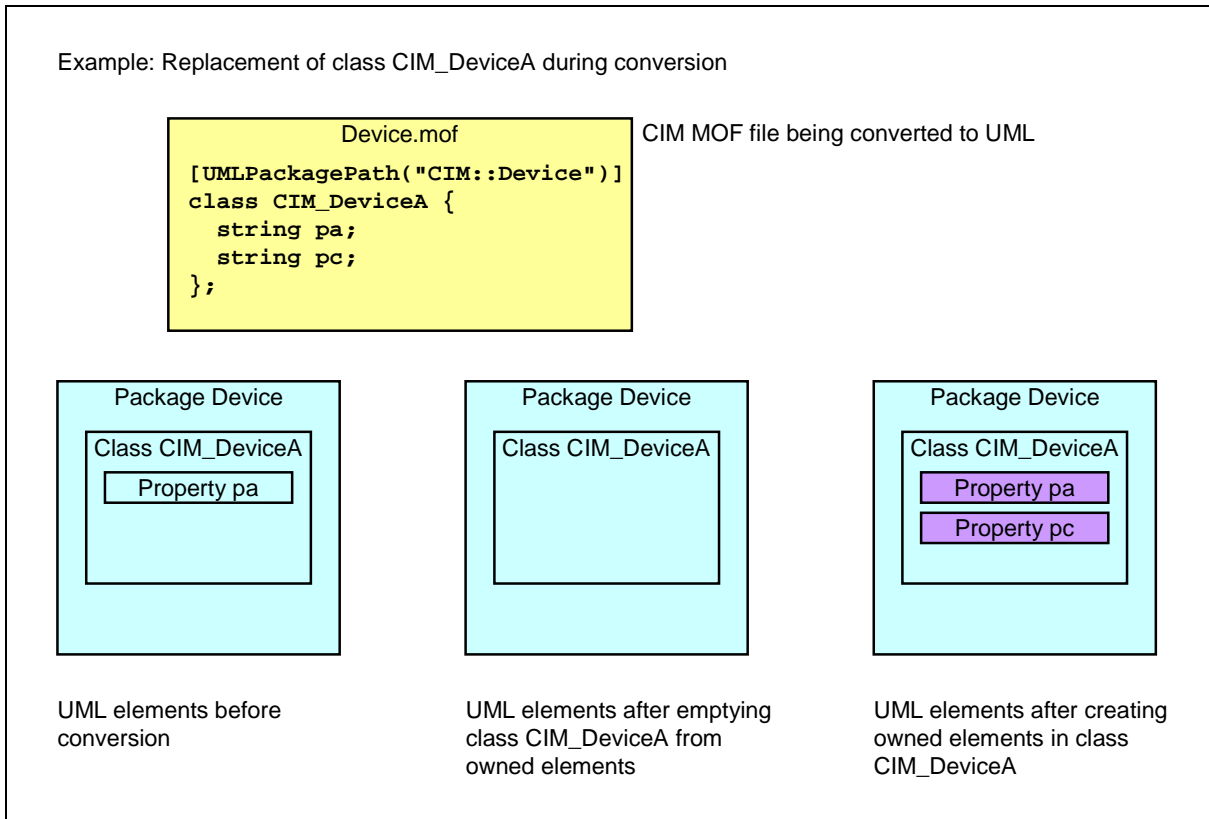
476

Figure 1 – Example for Package Mapping

477 In that example, the *UMLPackagePath* qualifier is set for class *CIM\_DeviceA*, so its value is used as the  
 478 relative package path under the target package path. Since in the example, the target package path is  
 479 *CIMSchema*, *CIM\_DeviceA* is contained by package *Device* which is contained by package *CIM* which is  
 480 contained by package *CIMSchema*. For class *CIM\_DeviceB*, the *UMLPackagePath* qualifier is not set  
 481 (i.e., it is NULL), so the default rule for the *UMLPackagePath* qualifier requires using "CIM::default", and  
 482 so class *CIM\_DeviceB* is contained by package *default* which is contained by package *CIM* which is  
 483 contained by package *CIMSchema*.



484 Figure 2 shows a non-normative example for replacing an existing UML class while converting CIM MOF  
 485 to UML.



486

487 **Figure 2 – Example for Replacement of UML Class During Conversion of CIM MOF to UML**

488 In that example, the class *CIM\_DeviceA* that is converted to UML already exists. During the conversion,  
 489 the owned elements of the class (property *pa*) are deleted, and then new owned elements are created  
 490 according to the definition of the class in the CIM MOF file (properties *pa*, *pc*). The UML element for the  
 491 class *CIM\_DeviceA* is retained, and so all references to that class using identifiers created by the  
 492 underlying UML storage format continue to be valid.

493 Per [DSP0004](#), the schema name of CIM classes (e.g., "CIM" in "CIM\_LogicalElement") shall be used as  
 494 the first package segment in the *UMLPackagePath* qualifier for all CIM classes.

### 495 5.4 Mapping of CIM Classes

496 Each CIM class shall be mapped to a UML *Class* metaclass instance. The CIM inheritance hierarchy shall  
 497 be maintained.

498 Table 2 defines the mapping for the attributes and associations of the UML *Class* metaclass.

Table 2 – UML Class Metaclass

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
isAbstract	Boolean	Classifier	Shall be mapped to the CIM <i>Abstract</i> qualifier as defined in 5.13.4.
isLeaf	Boolean	RedefinableElement	Shall be mapped to the CIM <i>Terminal</i> qualifier as defined in 5.13.4.
name	String	NamedElement	Shall be the name of the CIM class, including the CIM schema name.
visibility	VisibilityKind	PackagableElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
nestedClassifier	Classifier [0]	Class	Compliance value: no associated elements.
ownedAttribute	Property [*]	Class	Shall own and shall be associated with the <i>Property</i> metaclass instances to which the CIM properties are mapped, as defined in 5.7. The order of elements in <i>ownedAttribute</i> shall match the order of properties in the CIM class.
ownedOperation	Operation [*]	Class	Shall own and shall be associated with the <i>Operation</i> metaclass instances to which the CIM methods are mapped, as defined in 5.9. The order of elements in <i>ownedOperation</i> shall match the order of methods in the CIM class.
generalization	Generalization [0..1]	Classifier	If this CIM class has a superclass, shall own and shall be associated with a <i>Generalization</i> metaclass instance, as defined in 5.16.6, whose <i>general</i> association shall be associated with the <i>Class</i> metaclass instance to which the CIM superclass of this class is mapped. If this CIM class has no superclass, shall not have any associated elements.
redefinedClassifier	Classifier [0]	Classifier	Compliance value: no associated elements.
package	Package	Type (since UML 2.1)	Shall be mapped as defined in 5.3.
elementImport	ElementImport [0]	Namespace	Compliance value: no associated elements.
ownedRule	Constraint [*]	Namespace	Shall be mapped to "inv:" constraints of the CIM <i>ClassConstraint</i> qualifier of the CIM class, as defined in 5.13.4.
packageImport	PackageImport [0]	Namespace	Compliance value: no associated elements.
ownedComment	Comment [0..1]	Element	Shall be mapped to CIM <i>Description</i> qualifier of the CIM class, as defined in 5.13.4.

500 **5.5 Mapping of CIM Indications**

501 Each CIM indication shall be mapped to a UML *Class* metaclass instance. The CIM inheritance hierarchy  
 502 shall be maintained.

503 Table 2 defines the mapping for the attributes and associations of the UML *Class* metaclass.

504 **5.6 Mapping of CIM Associations**

505 Each CIM association class shall be mapped to a UML *AssociationClass* metaclass instance with owned  
 506 association ends. The CIM inheritance hierarchy shall be maintained.

507 Table 3 defines the mapping for the attributes and associations of the UML *AssociationClass* metaclass.

508 NOTE: The color and other graphical appearance of the association link are not normatively defined in this document,  
 509 because the UML Superstructure Specification does not allow specifying the graphical appearance. It is  
 510 recommended that tools apply the DMTF conventions for the color of association links (i.e., red for associations and  
 511 green for aggregations and compositions).

512 **Table 3 – UML *AssociationClass* Metaclass**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
all attributes from the CIM class mapping			
isDerived	Boolean	Association	Compliance value: false (UML default)
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
all associations from the CIM class mapping			
memberEnd	Property [2..*]	Association	Shall be associated with the <i>Property</i> metaclass instances to which the CIM references in this CIM association class are mapped, as defined in 5.8. The order of elements in <i>memberEnd</i> does not need to match the order in <i>ownedEnd</i> <sup>1</sup> .
ownedEnd	Property [2..*]	Association	Shall own and shall be associated with the <i>Property</i> metaclass instances to which the CIM references in this CIM association class are mapped, as defined in 5.8. The order of elements in <i>ownedEnd</i> shall match the order of the CIM references in the CIM association class.
navigableOwnedEnd	Property [2..*]	Association	Shall be associated with the <i>Property</i> metaclass instances to which the CIM references in this CIM association class are mapped, as defined in 5.8. The order of elements in <i>navigableOwnedEnd</i> does not need to match the order in <i>ownedEnd</i> <sup>2</sup> .

513 Notes: <sup>1</sup> UML infrastructures are supposed to set the *memberEnd* elements automatically as a result of setting any elements in  
 514 *ownedEnd*, in order to automatically satisfy the constraint that *ownedEnd* subsets *memberEnd*.

515 <sup>2</sup> In UML2 when association ends are owned by the association (as defined by this document), navigability of the  
 516 association is determined by the presence of the UML *Properties* that are associated by *ownedEnd*, in the  
 517 *navigableOwnedEnd* association. Note that while UML navigability cannot be represented in CIM, navigability is still  
 518 required to be set by valid CIM models.

519 **5.7 Mapping of CIM Properties**

520 Each CIM property defined in a CIM class shall be mapped to a UML *Property* metaclass instance  
 521 associated with the UML *Class* metaclass instance to which the CIM class is mapped. Covered properties  
 522 are treated like any other properties.

523 Table 4 defines the mapping for the attributes and associations of the UML *Property* metaclass.

524 **Table 4 – UML *Property* Metaclass Used to Map CIM Properties**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
aggregation	AggregationKind	Property	Compliance value: none (UML default).
isDerived	Boolean	Property	Compliance value: false (UML default).
isDerivedUnion	Boolean	Property	Compliance value: false (UML default).
isReadOnly	Boolean	Property	Shall be mapped to CIM <i>Write</i> qualifier as defined in 5.13.4.
isStatic	Boolean	Feature	Shall be mapped to CIM <i>Static</i> qualifier as defined in 5.13.4.
isLeaf	Boolean	RedefinableElement	Compliance value: false (UML default).
name	String	NamedElement	Shall be the name of the CIM property.
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private).
isOrdered	Boolean	MultiplicityElement	Shall be mapped to CIM <i>ArrayType</i> qualifier as defined in 5.13.4.
isUnique	Boolean	MultiplicityElement	Compliance value: false (UML default is true).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
association	Association [0]	Property	Compliance value: no associated elements.
owningAssociation	Association [0]	Property	Compliance value: no associated elements.
datatype	DataType [0]	Property	Compliance value: no associated elements (because this property is not owned by a structured datatype).
defaultValue	ValueSpecification [0..1]	Property	If the CIM property has no default value (i.e., an implied default value of NULL) or a specified default value of NULL, this association shall have either no associated elements or it shall own and shall be associated with a <i>LiteralNull</i> metaclass instance.  Otherwise, shall own and shall be associated with a subclass of the abstract <i>LiteralSpecification</i> metaclass that matches the type of the CIM property as defined in Table 8 (i.e., <i>LiteralBoolean</i> , <i>LiteralString</i> , <i>LiteralInteger</i> ). If <i>defaultValue</i> has no associated elements, this shall be interpreted as a default value of NULL.
redefinedProperty	Property [0..1]	Property	Mapped to the CIM <i>Override</i> qualifier as defined in 5.13.4.
subsettingProperty	Property [0]	Property	Compliance value: no associated elements.
associationEnd	Property [0]	Property	Compliance value: no associated elements.

525

UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
qualifier	Property [0]	Property	Compliance value: no associated elements.
type	Type	TypedElement	Shall be mapped as defined in Table 8.
lowerValue	ValueSpecification [0..1]	MultiplicityElement	<p>If the property has no array type, lowerValue either shall have no associated elements (implying the UML default value of "1"), or it shall own and shall be associated with a LiteralInteger metaclass instance whose value attribute is set to "1".</p> <p>Otherwise, lowerValue shall own and shall be associated with a LiteralInteger metaclass instance whose value attribute is set as follows: To "0", if the CIM property has array type of variable length; To the array size, if the CIM property has array type of fixed length.</p> <p>Note that the CIM qualifier Required does not define the presence of properties; it only defines whether or not a property can be NULL.</p>
upperValue	ValueSpecification [0..1]	MultiplicityElement	<p>If the property has no array type, upperValue either shall have no associated elements (implying the UML default value of "1"), or it shall own and shall be associated with a LiteralUnlimitedNatural metaclass instance whose value attribute is set to "1".</p> <p>Otherwise, upperValue shall own and shall be associated with a LiteralUnlimitedNatural metaclass instance whose value attribute is set as follows: To "**", if the CIM property has array type of variable length; To the array size, if the CIM property has array type of fixed length.</p>
ownedComment	Comment [0..1]	Element	Shall be mapped to CIM Description qualifier of the CIM property, as defined in 5.13.4.
class (since UML 2.1)	Class	Property	Shall be owned by and associated with the Class metaclass instance to which the CIM class defining the property is mapped, as defined in 5.4.
Stereotype Property	UML Type	Defined in Stereotype	CIM Mapping
ClassReferenceType	Enumeration CIM_ClassReferenceType_Enum	CIM_ClassReferenceType	Shall be mapped as defined in Table 8.

526 **5.8 Mapping of CIM References**

527 Each CIM reference defined in a CIM association class shall be mapped to a UML *Property* metaclass  
 528 instance associated with the UML *AssociationClass* metaclass instance to which the CIM association  
 529 class is mapped.

530 The mapping for CIM references used as method parameters is defined in 5.10.

531 Table 5 defines the mapping for the attributes and associations of the UML *Property* metaclass.

532

Table 5 – UML *Property* Metaclass Used to Map CIM References

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
All attributes from the CIM property mapping unless redefined in the remainder of this table.			
aggregation	AggregationKind	Property	Shall be mapped to CIM <i>Association</i> , <i>Aggregation</i> , <i>Composition</i> and <i>Aggregate</i> qualifiers as defined in 5.13.4.
isStatic	Boolean	Feature	Compliance value: false (UML default).
isOrdered	Boolean	MultiplicityElement	Compliance value: false (UML default).
isUnique	Boolean	MultiplicityElement	Compliance value: true unless there are non-reference key properties defined in the same class (UML default is true).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
All associations from the CIM property mapping unless redefined in the remainder of this table.			
association	Association	Property	Shall be associated with the <i>AssociationClass</i> metaclass instance to which the CIM association class that defines the CIM reference is mapped.
owningAssociation	Association	Property	Shall be associated with the <i>AssociationClass</i> metaclass instance to which the CIM association class that defines the CIM reference is mapped.
type	Type	TypedElement	Shall be mapped as defined in Table 8.
lowerValue	ValueSpecification	MultiplicityElement	Shall be mapped to the CIM <i>Min</i> qualifier as defined in 5.13.4.
upperValue	ValueSpecification	MultiplicityElement	Shall be mapped to the CIM <i>Max</i> qualifier as defined in 5.13.4.

## 533 5.9 Mapping of CIM Methods

534 Each CIM method shall be mapped to a UML *Operation* metaclass instance associated with the UML  
535 *Class* metaclass instance to which the CIM class owning the method is mapped.

536 While the method return value in CIM is part of the *Method* meta element, it is not in UML. In UML, an  
537 instance of the *Parameter* metaclass with its *direction* attribute set to *return* is used to represent a method  
538 return value. Therefore, the mapping of the return value of CIM methods is defined in 5.10.

539 Table 6 defines the mapping for the attributes and associations of the UML *Operation* metaclass.

540

Table 6 – UML *Operation* Metaclass Used to Map CIM Methods

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
isQuery	Boolean	Operation	Compliance value: false (UML default).
isStatic	Boolean	Feature	Shall be mapped to the CIM <i>Static</i> qualifier as defined in 5.13.4.
isLeaf	Boolean	RedefinableElement	Compliance value: false (UML default).
name	String	NamedElement	Shall be the name of the CIM method, without signature.
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private).

UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
class	Class	Operation	Shall be associated with the UML <i>Class</i> metaclass instance to which the CIM class is mapped.
bodyCondition	Constraint [0]	Operation	Compliance value: no associated elements.
ownedParameter	Parameter [*]	Operation	Shall own and shall be associated with the <i>Parameter</i> metaclass instances to which the CIM method parameters and the return value are mapped, as defined in 5.10. The order of elements in <i>ownedParameter</i> shall match the order of parameters in the CIM class.
postcondition	Constraint [*]	Operation	Shall be mapped to "post:" constraints of CIM <i>MethodConstraint</i> qualifier of the CIM method, as defined in 5.13.4.
precondition	Constraint [*]	Operation	Shall be mapped to "pre:" constraints of CIM <i>MethodConstraint</i> qualifier of the CIM method, as defined in 5.13.4.
raisedException	Type [0]	Operation	Compliance value: no associated elements.
redefinedOperation	Operation [0..1]	Operation	Shall be mapped to the CIM <i>Override</i> qualifier as defined in 5.13.4.
elementImport	ElementImport [0]	Namespace	Compliance value: no associated elements.
ownedRule	Constraint [*]	Namespace	Shall be mapped to "body:" constraints of CIM <i>MethodConstraint</i> qualifier of the CIM method, as defined in 5.13.4.
packageImport	PackageImport [0]	Namespace	Compliance value: no associated elements.
ownedComment	Comment [0..1]	Element	Shall be mapped to the CIM <i>Description</i> qualifier of the CIM method, as defined in 5.13.4.

541 **5.10 Mapping of CIM Parameters and CIM Return Values**

542 Each parameter and the return value of CIM methods shall be mapped to a UML *Parameter* metaclass  
 543 instance associated with the UML *Operation* metaclass instance to which the CIM method is mapped.

544 Table 7 defines the mapping for the attributes and associations of these UML *Parameter* metaclass, for  
 545 both the cases of parameters and return values. It applies both to reference types and non-reference  
 546 types.

547 **Table 7 – UML *Parameter* Metaclass Used to Map CIM Parameters and CIM Return Values**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
direction	ParameterDirection Kind	Parameter	For CIM parameters: shall be mapped to the CIM <i>In</i> and <i>Out</i> qualifiers of the CIM parameter, as defined in 5.13.4.  For the CIM return value: shall be mapped to the <i>ParameterDirectionKind</i> enumeration value "return".
isOrdered	Boolean	MultiplicityElement	Shall be mapped to the CIM <i>ArrayType</i> qualifier as defined in 5.13.4.
isUnique	Boolean	MultiplicityElement	Compliance value: false (UML default is true).

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
name	String	NamedElement	For CIM parameters: shall be the name of the CIM parameter. For the CIM return value: should be the string "ReturnValue".
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
defaultValue	ValueSpecification [0]	Parameter	Compliance value: no associated elements.
type	Type	TypedElement	Shall be mapped as defined in Table 8.
lowerValue	ValueSpecification	MultiplicityElement	If the parameter has no array type, <i>lowerValue</i> either shall have no associated elements (implying the UML default value of "1"), or it shall own and shall be associated with a <i>LiteralInteger</i> metaclass instance whose <i>value</i> attribute is set to "1".  Otherwise, shall own and shall be associated with a <i>LiteralInteger</i> metaclass instance whose <i>value</i> attribute is set as follows: To "0", if the CIM property has array type of variable length; To the array size, if the CIM property has array type of fixed length.  Note: The CIM qualifier <i>Required</i> does not define the presence of parameters; it only defines whether a parameter can be NULL.
upperValue	ValueSpecification	MultiplicityElement	If the parameter has no array type, <i>upperValue</i> either shall have no associated elements (implying the UML default value of "1"), or it shall own and shall be associated with a <i>LiteralUnlimitedNatural</i> metaclass instance whose <i>value</i> attribute is set to "1".  Otherwise, shall own and shall be associated with a <i>LiteralUnlimitedNatural</i> metaclass instance whose <i>value</i> attribute is set as follows: To "*", if the CIM property has array type of variable length; To the array size, if the CIM property has array type of fixed length
ownedComment	Comment [0..1]	Element	Shall be mapped to the CIM <i>Description</i> qualifier of the CIM parameter, as defined in 5.13.4.
Stereotype Property	UML Type	Defined in Stereotype	CIM Mapping
ClassReferenceType	Enumeration CIM_ClassReferenceType_Enum	CIM_ClassReferenceType	Shall be mapped as defined in Table 8.



548 **5.11 Mapping of CIM Datatypes**

549 This subclause normatively defines how CIM datatypes are mapped to UML constructs.

550 Table 8 defines the mapping of CIM datatypes. Any *Data Type* instances defined in this table shall be  
 551 owned by a UML Type Library, as defined in 5.21.3.

552 The UML construct specified in the "UML construct for *type* association" column shall be used for the *type*  
 553 association in the UML metaclass instance using the type.

554 Since the [CIM Infrastructure Specification](#) defines that CIM datatypes have no subtype relationships  
 555 between each other, the *Data Type* instances defined in Table 8 shall have no supertype.

556 The "UML metaclass used for values" column defines the UML metaclass that shall be used for values of  
 557 this CIM datatype in a UML user model, for example in default values or instance property values.

558 **Table 8 – Mapping of CIM Datatypes**

CIM Datatype	UML Construct for <i>type</i> Association	UML Metaclass Used for Values
uint8	<i>Data Type</i> instance named "uint8"	LiteralInteger <sup>1</sup>
sint8	<i>Data Type</i> instance named "sint8"	LiteralInteger <sup>1</sup>
uint16	<i>Data Type</i> instance named "uint16"	LiteralInteger <sup>1</sup>
sint16	<i>Data Type</i> instance named "sint16"	LiteralInteger <sup>1</sup>
uint32	<i>Data Type</i> instance named "uint32"	LiteralInteger <sup>1</sup>
sint32	<i>Data Type</i> instance named "sint32"	LiteralInteger <sup>1</sup>
uint64	<i>Data Type</i> instance named "uint64"	LiteralInteger <sup>1</sup>
sint64	<i>Data Type</i> instance named "sint64"	LiteralInteger <sup>1</sup>
string	<i>Data Type</i> instance named "string"	LiteralString <sup>1</sup>
string with <i>EmbeddedInstance</i> qualifier	UML <i>Class</i> instance to which the CIM class specified as a value of <i>EmbeddedInstance</i> , is mapped. In addition, the <i>ClassReferenceType</i> stereotype property of the UML metaclass instance to which the qualified CIM element is mapped, shall be set to <i>ByValue</i> .	See "UML construct" column
string [ ] with <i>Octetstring</i> qualifier	Mapped like an array of <i>Data Type</i> "octetstring". See the last row in the table for how arrays are mapped.	LiteralString <sup>1</sup>
uint8 [ ] with <i>Octetstring</i> qualifier	<i>Data Type</i> instance named "octetstring"	LiteralString <sup>1</sup>
boolean	<i>Data Type</i> instance named "boolean"	LiteralBoolean <sup>1</sup>
real32	<i>Data Type</i> instance named "real32"	LiteralInteger <sup>1</sup>
real64	<i>Data Type</i> instance named "real64"	LiteralInteger <sup>1</sup>
datetime	<i>Data Type</i> instance named "datetime"	LiteralString <sup>1</sup>
<classname> ref	UML <i>Class</i> instance to which the CIM class that is referenced in the CIM reference is mapped. In addition, for parameters, the <i>ClassReferenceType</i> stereotype property of the UML <i>Parameter</i> to which the CIM parameter is mapped, shall be set to <i>ByReference</i> .	Ends of association instances
char16	<i>Data Type</i> instance named "char16"	LiteralString <sup>1</sup>

CIM Datatype	UML Construct for <i>type</i> Association	UML Metaclass Used for Values
arrays	The CIM datatype of the array elements is mapped as defined in this table, and the array characteristic is mapped to the <i>lowerValue</i> and <i>upperValue</i> associations of the UML <i>Property</i> and <i>Parameter</i> metaclasses, as defined in 5.7 and 5.10.	See "UML construct" column

559 Note: <sup>1</sup>The specified UML metaclass is preferred for simplicity and interoperability. All other valid representations of such values  
 560 as defined in the [UML Superstructure Specification](#) may be used in addition.

561 **5.12 Mapping of CIM Instances**

562 Each modeled CIM instance shall be mapped to a UML *InstanceSpecification* metaclass instance. A  
 563 modeled CIM instance is one that is owned by a UML user model. This includes CIM instances defined in  
 564 CIM MOF and imported into UML.

565 NOTE: Modeled CIM instances do not necessarily represent the run-time CIM instances in a CIM server. In case  
 566 where the modeled instance resulted from importing a CIM MOF definition of a CIM instance into UML, and where the  
 567 same CIM MOF definition is imported into a CIM server, the modeled instance represents at least the initial state of a  
 568 run-time instance. However, modeled instances can also be used to describe use cases in UML, and in such cases  
 569 typically only a subset of the properties is described, and the run-time instances cannot be deducted from that. Also,  
 570 there may be multiple modeled instances that represent different points in time for the same run-time instance.

571 Table 9 defines the mapping for the attributes and associations of the UML *InstanceSpecification*  
 572 metaclass.

573 **Table 9 – UML *InstanceSpecification* Metaclass Used for Modeled CIM Instances**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
name	String	NamedElement	No compliance value is defined; extension point.
visibility	VisibilityKind	PackagableElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
classifier	Classifier [1]	InstanceSpecification	Shall be associated with the UML <i>Class</i> metaclass instance to which the CIM creation class of the modeled CIM instance is mapped.
slot	Slot [*]	InstanceSpecification	Shall own and shall be associated with an instance of the UML <i>Slot</i> metaclass for each CIM property that has a value defined in the modeled CIM instance, as defined in Table 10.
specification	ValueSpecification [0]	InstanceSpecification	Compliance value: no associated elements.
ownedComment	Comment [0..1]	Element	No compliance value is defined; extension point.

574 Table 10 defines the mapping for the attributes and associations of the UML *Slot* metaclass referenced  
 575 from Table 9.

576 **Table 10 – UML *Slot* Metaclass Used for Property Values of Modeled CIM Instances**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
No attributes.			
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
definingFeature	StructuralFeature [1]	Slot	Shall be associated with the UML <i>Property</i> metaclass instance to which the CIM property is mapped.
owningInstance	InstanceSpecification [1]	Slot	Shall be associated with the UML <i>InstanceSpecification</i> metaclass instance to which the CIM instance that has the CIM property value is mapped.
value	ValueSpecification [*]	Slot	Shall own and shall be associated with instances of a subclass of the abstract <i>LiteralSpecification</i> metaclass that matches the type of the CIM property as defined in Table 8 (i.e., <i>LiteralBoolean</i> , <i>LiteralString</i> , <i>LiteralInteger</i> ) or <i>LiteralNull</i> to represent the CIM NULL value. There shall be one instance for each value (i.e., for non-arrays, there is one instance, and for arrays, there is one instance for each array element that has a value).
ownedComment	Comment [0..1]	Element	No compliance value is defined; extension point.

577 **5.13 Mapping of CIM Qualifiers**

578 **5.13.1 Overview**

579 This subclause gives an overview of the mapping of CIM qualifier types and CIM qualifier values to UML  
 580 constructs. Note that CIM qualifier types are sometimes called "qualifier declarations", but we use the  
 581 term "qualifier types" in this document.

582 In CIM, there is a distinction between qualifier types and qualifier values: The definition of a qualifier type  
 583 is needed in order to define a qualifier value of that type on a CIM element. Since the definition of qualifier  
 584 types is dynamic in CIM, this document defines a UML mapping for the qualifier types in order to  
 585 represent the definition of a qualifier type, as well as a UML mapping for the qualifier values.

586 The set of DMTF defined qualifier types is defined in each release of the CIM Schema. The [CIM](#)  
 587 [Infrastructure Specification](#) specifies all DMTF qualifier types at a particular point in time, but typically the  
 588 CIM Schema is on shorter release cycle than the [CIM Infrastructure Specification](#), and so the CIM  
 589 Schema ends up to be the normative definition of the set of DMTF defined qualifier types.

590 The set of CIM qualifiers is vendor extensible, allowing vendors to define additional qualifier types. In a  
 591 particular vendor extension schema, the DMTF defined qualifier types and the vendor defined qualifier  
 592 types make up the complete set of qualifier types that is to be mapped to UML.

593 These qualifier types (i.e., the qualifier type definitions) are mapped to UML as defined in 5.13.2, for all  
 594 qualifier types.

595 The qualifier values are mapped to UML using two mapping approaches, depending on the qualifier type:

596 • **Direct Qualifier Mapping**

597 Subclause 5.13.4 defines the mapping rules for values of a specific set of qualifier types. That  
598 subclause covers the subset of CIM defined standard and optional qualifier types that are  
599 mapped to specific UML features. A tool that imports CIM MOF into a UML user model needs to  
600 recognize these qualifiers by their name in order to implement the specific mapping rules.

601 • **Generic Qualifier Mapping**

602 Subclause 5.13.3 defines the mapping rules for values of all qualifier types that are not in the  
603 specific set covered by the direct qualifier mapping. A tool that imports CIM MOF into a UML  
604 user model can implement generic support for this mapping approach without requiring  
605 knowledge about the specific qualifier types; hence the mapping approach allows to deal with  
606 an unknown set of qualifiers without having to update this document nor the mapping support in  
607 tools.

608 The inheritance behavior of CIM qualifiers and the inheritance behavior of the UML constructs to which  
609 they are mapped are somewhat different, as explained in the following paragraphs.

610 The inheritance behavior of CIM qualifiers is defined in the [CIM Infrastructure Specification](#) and is  
611 repeated here briefly:

- 612 • CIM qualifier values do not need to be defined explicitly on each CIM element.
- 613 • However, all qualifiers have an effective value on each CIM element, as follows:
  - 614 – If a value for a qualifier is defined explicitly (for example, "Write=true" or "Write") on the  
615 CIM element, the effective qualifier value is the value defined there.
  - 616 – If no value for a qualifier is defined explicitly (for example, "", which means that "Write" is  
617 not specified) on the CIM element, the effective qualifier value is determined as follows:
    - 618 • Qualifiers with flavor *ToSubclass* inherit their effective value along the inheritance  
619 chain. Classes that have no superclass (i.e., top classes) and CIM elements within  
620 such classes obtain their effective value from the default value defined in the  
621 declaration of the qualifier type.
    - 622 • Qualifiers with flavor *Restricted* obtain their effective value from the default value  
623 defined in the declaration of the qualifier type.

624 The inheritance behavior of the UML constructs to which the CIM qualifier values are mapped, is as  
625 follows:

- 626 • Some UML constructs are required to be present at each level of a class inheritance chain (for  
627 example, the *isAbstract* attribute). For these constructs, there is no automatic propagation of  
628 values along the inheritance chain, i.e., each value stands on its own and a mapping rule needs  
629 to cover each level of the inheritance chain even if the qualifier value is not defined at each of  
630 those levels.
- 631 • Some UML constructs are optionally present at each level of a class inheritance chain (for  
632 example, stereotypes or associated UML metaclass instances). These UML constructs are not  
633 inherited by subclasses in a UML user model.

634 The following general principles are followed in order to map these different inheritance behaviors:

- 635 1) For CIM qualifiers mapped to UML constructs that are required to be present at each level of an  
636 inheritance chain, the qualifier value represented in the UML construct is the effective qualifier  
637 value for that level in the inheritance chain. This leads to redundant qualifier values in the UML  
638 user model even if they were not redundant in the CIM model, but that is unavoidable because  
639 the UML constructs are required to be present.

640 2) For CIM qualifiers mapped to UML constructs that are optionally present at each level of an  
 641 inheritance chain, the UML construct is present if and only if the effective qualifier value for that  
 642 level in the inheritance chain is different from the default value (for *Restricted* qualifiers) or the  
 643 inherited value (for *ToSubclass* qualifiers). In other words, the presence of the UML constructs  
 644 is minimized as much as possible, removing any redundancy, even if the value was redundantly  
 645 defined in the CIM model. If the UML construct is present, the qualifier value represented in the  
 646 UML construct is the effective qualifier value for that level in the inheritance chain.

647 During the course of editing a UML user model with a UML editor, optional UML constructs representing  
 648 qualifiers may be present even if their value is the same as the inherited value. A tool can provide a  
 649 means to remove such redundancies.

650 Because the second principle avoids redundant qualifier value definitions, a CIM MOF that contained  
 651 such redundancies may be different from the CIM MOF that is created when the original CIM MOF is  
 652 imported into a UML user model and exported again. Semantically, both versions are the same however  
 653 and no information is lost. There are a few qualifiers in CIM for which redundant values are used  
 654 commonly even though this is not mandated by the CIM Infrastructure Specification. This document  
 655 accommodates these special cases.

656 These general principles are reflected by the normative mapping rules in the subsequent subclauses.

657 **5.13.2 Mapping of Qualifier Types**

658 NOTE: The general principle for the mapping of CIM qualifier types (i.e., the qualifier declarations) to UML is that the  
 659 UML construct to which they are mapped is able to represent the qualifier type itself as a self-contained entity,  
 660 independently of any usages of the qualifier type in qualifier values.

661 Each CIM qualifier type shall be mapped to an instance of the UML *Stereotype* metaclass. These  
 662 stereotypes are called "Qualifier Type Stereotypes" in this document. The Qualifier Type Stereotypes are  
 663 owned by a UML profile as defined in 5.21.

664 Table 11 defines the mapping for the attributes and associations of the UML *Stereotype* metaclass for this  
 665 purpose.

666 NOTE: The UML *Stereotype* metaclass has mostly attributes and associations from the UML Infrastructure Library as  
 667 defined in the [UML Infrastructure Specification](#).

668 **Table 11 – UML Stereotype Metaclass Used as Qualifier Type Stereotype**

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
isAbstract	Boolean	Class	Compliance value: true. Note that the qualifier type stereotypes have no extensions defined and are not applied directly. Instead, they are inherited into the qualifier scope stereotypes defined in Table 15.
isLeaf	Boolean	Not defined in the <a href="#">CIM Infrastructure Specification</a> at this point	If supported by the UML tool then compliance value: false.

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
name	String	NamedElement	Shall be the name of the CIM qualifier type, using the following syntax: "CIM_Qualifier_" qualifier-name Where <i>qualifier-name</i> is the qualifier name as defined in the CIM MOF declaration of the qualifier type. The lexical case of the name shall be preserved. Note that this rule does not contradict the rule that names in CIM are compared case-insensitively. Note that the CIM defined qualifiers as described in the <a href="#">CIM Infrastructure Specification</a> use all-upper-case characters, while the CIM MOF declaration in the CIM Schema uses mixed case.
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private)
UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedAttribute	Property [1]	Class	Shall own and shall be associated with a single UML <i>Property</i> metaclass instance representing the definitional aspects of the qualifier value, as defined in Table 12.
ownedOperation	Operation [0]	Class	Compliance value: no associated elements.
superClass	Class [0]	Class	Compliance value: no associated elements.
elementImport	ElementImport [0]	Namespace	Compliance value: no associated elements.
packageImport	PackageImport [0]	Namespace	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.
icon	Image [*]	UML <i>Stereotype</i> metaclass	No compliance value is defined; extension point.

669 Each Qualifier Type Stereotype shall own a single property representing the definitional aspects of the  
 670 qualifier value, as defined in Table 12.

671 NOTE: Even though the *Stereotype.ownedAttribute* association is owned by a metaclass contained in a package from  
 672 the *InfrastructureLibrary* package, the *Property* metaclass it is associated with is from the *UML::Classes* package.

673 The set of all qualifier types defined in the "CIM" profile shall be determined by enumerating any  
 674 stereotypes defined in the "CIM" profile that have a name starting with "CIM\_Qualifier\_".

675 A stereotype *CIM\_QualifierType* (defined further down) shall be applied to this property, adding some  
 676 attributes needed for qualifiers. Table 12 also defines how these additional attributes are mapped.

677 **Table 12 – UML *Property* Metaclass Used for Definitional Aspects of CIM Qualifier Value Within**  
 678 **Qualifier Type Stereotype**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
aggregation	AggregationKind	Property	Compliance value: none (UML default).
isDerived	Boolean	Property	Compliance value: false (UML default).
isDerivedUnion	Boolean	Property	Compliance value: false (UML default).
isReadOnly	Boolean	Property	Compliance value: false (UML default).

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
isStatic	Boolean	Feature	Compliance value: false (UML default).
isLeaf	Boolean	RedefinableElement	Compliance value: false (UML default).
name	String	NamedElement	Shall be the name of the CIM qualifier as defined in CIM MOF, preserving the lexical case.
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private).
isOrdered	Boolean	MultiplicityElement	For CIM qualifiers that are arrays: compliance value: true. The order of the array entries shall be preserved. For CIM qualifiers that are not arrays: no compliance value defined. Note that <i>isOrdered</i> is meaningless for non-arrays.
isUnique	Boolean	MultiplicityElement	Compliance value: false (UML default is true).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
association	Association [0]	Property	Compliance value: no associated elements.
owningAssociation	Association [0]	Property	Compliance value: no associated elements.
datatype	DataType [0]	Property	Compliance value: no associated elements (because this property is not owned by a structured datatype).
defaultValue	ValueSpecification [0..1]	Property	If the CIM qualifier type has no default value (i.e., an implied default value of NULL) or a specified default value of NULL, this association shall have either no associated elements or it shall own and shall be associated with a <i>LiteralNull</i> metaclass instance.  Otherwise, shall own and shall be associated with a subclass of the abstract <i>LiteralSpecification</i> metaclass that matches the type of the CIM qualifier as defined in Table 8 (i.e., <i>LiteralBoolean</i> , <i>LiteralString</i> , <i>LiteralInteger</i> ) or <i>LiteralNull</i> to represent the CIM NULL value. If <i>defaultValue</i> has no associated elements, this shall be interpreted as a default value of NULL.
redefinedProperty	Property [0]	Property	Compliance value: no associated elements.
subsettingProperty	Property [0]	Property	Compliance value: no associated elements.
associationEnd	Property [0]	Property	Compliance value: no associated elements.
qualifier	Property [0]	Property	Compliance value: no associated elements.
type	Type	TypedElement	Shall be mapped as defined in Table 8.
lowerValue	ValueSpecification [0..1]	MultiplicityElement	Shall own and shall be associated with a <i>LiteralInteger</i> metaclass instance whose <i>value</i> attribute is set to "0". For a discussion of the reasons for this mapping, refer to the notes in 5.13.3.

UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
upperValue	ValueSpecification [0..1]	MultiplicityElement	If the property has no array type, <i>upperValue</i> either shall have no associated elements (implying the UML default value of "1"), or it shall own and shall be associated with a <i>LiteralUnlimitedNatural</i> metaclass instance whose <i>value</i> attribute is set to "1".  Otherwise, <i>upperValue</i> shall own and shall be associated with a <i>LiteralUnlimitedNatural</i> metaclass instance whose <i>value</i> attribute is set as follows: To "*", if the CIM qualifier has array type of variable length; To the array size, if the CIM qualifier has array type of fixed length.
ownedComment	Comment [*]	Element	Currently, there is no description for qualifiers in CIM. Therefore: no compliance value defined, extension point.
class (since UML 2.1)	Class	Property	Shall be owned by and associated with the <i>Stereotype</i> metaclass instance owning the property.
Stereotype Property	UML Type	Defined in Stereotype	CIM Mapping
InheritanceRule	Enumeration CIM_QualifierInheritanceRule	CIM_QualifierType	Shall be mapped to the inheritance flavors of the CIM qualifier, as defined in Table 13.
Translatable	Boolean	CIM_QualifierType	Shall be the <i>Translatable</i> flavor of the CIM qualifier.

680 The stereotype *CIM\_QualifierType* shall be defined in the UML profile "CIMQualifierType" and shall have  
681 the Properties defined in Table 13.

682 **Table 13 – Properties of Stereotype *CIM\_QualifierType***

Name	Type	Default Value	Meaning
InheritanceRule	Enumeration CIM_QualifierInheritanceRule	ToSubclass EnableOverride	Inheritance flavor of the qualifier type.
Translatable	Boolean	False	<i>Translatable</i> flavor of the qualifier type.

683 The enumeration *CIM\_QualifierInheritanceRule* shall be defined in the UML profile "CIMQualifierType"  
684 and shall have the literals defined in Table 14.

685 **Table 14 – Literals of Enumeration *CIM\_QualifierInheritanceRule***

Literal Name	Meaning
ToSubclassEnableOverride	Represents qualifier inheritance flavor <i>ToSubclass EnableOverride</i> .
ToSubclassDisableOverride	Represents qualifier inheritance flavor <i>ToSubclass DisableOverride</i> .
Restricted	Represents qualifier inheritance flavor <i>Restricted</i> .

686 The following stereotypes shall be defined and owned by the UML profile "CIM". These stereotypes  
687 are called "Qualifier Scope Stereotypes" in this document because there is one stereotype for each  
688 qualifier scope. They form an inheritance tree. Table 15 defines these stereotypes.



689

**Table 15 – Qualifier Scope Stereotypes**

Stereotype Name	Abstract	Base Stereotypes	Meaning
CIM_Any_Qualifiers	Yes	All qualifier type stereotypes of qualifiers with scope <i>Any</i> .	Represents the qualifiers with scope <i>Any</i> . In CIM, scope <i>Any</i> is a shorthand for all defined scopes.
CIM_Class_Qualifiers	Yes	All qualifier type stereotypes of qualifiers with scope <i>Class</i> but not <i>Any</i> , and <i>CIM_Any_Qualifiers</i> .	Represents the qualifiers with scope <i>Class</i> (including <i>Any</i> ).
CIM_Association_Qualifiers	Yes	All qualifier type stereotypes of qualifiers with scope <i>Association</i> but not <i>Any</i> , and <i>CIM_Any_Qualifiers</i> .	Represents the qualifiers with scope <i>Association</i> (including <i>Any</i> ).
CIM_Indication_Qualifiers	Yes	All qualifier type stereotypes of qualifiers with scope <i>Indication</i> but not <i>Any</i> , and <i>CIM_Any_Qualifiers</i> .	Represents the qualifiers with scope <i>Indication</i> (including <i>Any</i> ).
CIM_Property_Qualifiers	Yes	All qualifier type stereotypes of qualifiers with scope <i>Property</i> but not <i>Any</i> , and <i>CIM_Any_Qualifiers</i> .	Represents the qualifiers with scope <i>Property</i> (including <i>Any</i> ).
CIM_Reference_Qualifiers	Yes	All qualifier type stereotypes of qualifiers with scope <i>Reference</i> but not <i>Any</i> , and <i>CIM_Any_Qualifiers</i> .	Represents the qualifiers with scope <i>Reference</i> (including <i>Any</i> ).
CIM_Method_Qualifiers	Yes	All qualifier type stereotypes of qualifiers with scope <i>Method</i> but not <i>Any</i> , and <i>CIM_Any_Qualifiers</i> .	Represents the qualifiers with scope <i>Method</i> (including <i>Any</i> ).
CIM_Parameter_Qualifiers	Yes	All qualifier type stereotypes of qualifiers with scope <i>Parameter</i> but not <i>Any</i> , and <i>CIM_Any_Qualifiers</i> .	Represents the qualifiers with scope <i>Parameter</i> (including <i>Any</i> ).

690 **5.13.3 Generic Mapping of Qualifier Values**

691 This subclause defines a mapping of the qualifier values that shall be used for any CIM qualifier types  
 692 that are not covered in 5.13.4. In this document, these qualifier types are called the *generically mapped*  
 693 qualifier types.

694 NOTE: Through the mapping of the CIM qualifier types as defined in 5.13.2, each CIM element in a UML user model  
 695 has the capability to have qualifier values according to their scope. These qualifier values are represented by the  
 696 values of the qualifier properties inherited into the Meta Element Stereotype applied to the UML elements  
 697 representing CIM elements. Meta Element Stereotypes are defined in 5.15.

698 The [UML Superstructure Specification](#) does not currently define whether or not the values of stereotype  
 699 properties in applied stereotypes may be omitted. However, optionality of the property values is needed in  
 700 order to represent the optional presence of qualifier value definitions on CIM elements.

701 This document assumes that the values of stereotype properties may be omitted if the lower bound of  
 702 their multiplicity is 0.

703 The [UML Superstructure Specification](#) currently leaves some room for interpretation with respect to how  
 704 an applied stereotype is represented in UML. However, this document needs to define the mapping of  
 705 qualifier values to property values within applied stereotypes in terms of such a representation.

706 This document chooses the interpretation that an applied stereotype is represented as a separate UML  
 707 metaclass named after the stereotype name, which has an accordingly typed attribute for each property  
 708 owned by the defining stereotype.

709 Figure 6 shows an example of applied stereotypes with their attributes that represent the qualifier values.

710 Table 16 defines how the qualifier values for generically mapped qualifiers are represented using applied  
711 Meta Element Stereotypes.

712 **Table 16 – Applied Meta Element Stereotypes Used for CIM Qualifier Values**

UML attribute	UML type	Defined in UML metaclass	CIM Mapping
<qualifier-name>	<qualifier-type> [0..x]	(each Meta Element Stereotype)	For each CIM qualifier defined on the CIM element represented by the UML element the stereotype is applied to, this attribute shall have the value of the CIM qualifier. If a qualifier is not defined on the CIM element, the attribute shall not have a value.

713 Where: <qualifier-name> is the value of the *name* attribute of the stereotype property, as defined in Table 12.  
714 <qualifier-type> is a datatype according to the *type* association of the stereotype property, as defined in Table 12.  
715 [0..x] is the multiplicity of the stereotype property, as defined in Table 12.

716 Qualifier types declared with the CIM flavor *DisableOverride* shall not be overridden, as defined in 4.5.4  
717 (Class and Property Qualifiers) of the [CIM Infrastructure Specification](#).

718 As described in 5.13.1, each qualifier type that applies to a CIM element has an effective value of the  
719 qualifier on that CIM element, regardless of whether or not the qualifier is defined (i.e., set) on the CIM  
720 element.

721 The *BitMap* and *BitValues* qualifiers are intentionally mapped generically, because their usage in the CIM  
722 Schema 2.x is minimal and did not seem to warrant the effort to define a direct mapping.

723 As defined in the [CIM Infrastructure Specification](#), qualifiers with the *Translatable* flavor have related  
724 implicitly defined qualifier types with locale specific suffixes in the qualifier name. Such locale specific  
725 qualifiers are mapped generically, regardless of whether the related qualifier without locale suffix is  
726 mapped generically or directly. For example, the *Description* qualifier is mapped directly, and its related  
727 locale specific qualifier *Description\_de\_DE* is mapped generically.

728 Values of qualifiers with the *Translatable* flavor shall be interpreted using the effective value of the  
729 *ClassLocale* stereotype property of the UML metaclass instance to which the qualified CIM class or the  
730 CIM class owning the qualified element is mapped. Subclause 5.14 describes how to determine the  
731 effective value of this stereotype property.

#### 732 **5.13.4 Direct Mapping of Qualifier Values**

733 This subclause defines the mapping of the qualifier values of a specific set of CIM qualifier types that  
734 have semantically equivalent UML constructs, or for which specific UML stereotypes are defined. These  
735 qualifier types are called the *directly mapped* qualifier types in this document.

##### 736 **5.13.4.1 Abstract Qualifier**

737 The *isAbstract* attribute of any UML *Class* metaclass instance shall be equal to the effective value of the  
738 *Abstract* qualifier of the corresponding CIM class.

739 The *CIM::Abstract* stereotype property of the UML metaclass instance mapped to the qualified CIM  
740 element shall not be set.

##### 741 **5.13.4.2 Association, Aggregation, Composition, Aggregate Qualifiers**

742 As defined in 5.6, each CIM association class is mapped to a UML *AssociationClass* metaclass instance,  
743 and each CIM reference within a CIM association class is mapped to a UML *Property* metaclass instance.  
744 The UML *Property* metaclass has an *aggregation* attribute which specifies the type of aggregation that  
745 applies to the association end represented by that UML Property.

746 The value of the *aggregation* attribute of any UML *Property* metaclass instance representing a CIM  
 747 reference shall be mapped to the effective values of these qualifiers on the corresponding CIM reference,  
 748 as defined in Table 17.

749 **Table 17 – Mapping of Association/Aggregation/Composition/Aggregate Qualifiers**

Kind of CIM Association Class	Value of UML <i>aggregation</i> Attribute	
	On Aggregate Reference	On Other Reference
Association	None	None
Aggregation	None	Shared
Composition	None	Composite

750 The "Aggregate reference" is the CIM reference in a CIM association class that has an effective value of  
 751 *true* for its *Aggregate* qualifier.

752 NOTE 1: In CIM, the *Aggregate* qualifier is set on the association end that is the "whole". In UML, the diamond  
 753 adornment for aggregation is also set on the association end that is the "whole", while the "shared" value of the UML  
 754 *aggregation* attribute is set on the association end that is the "part".

755 NOTE 2: In UML, aggregations and compositions are restricted to an arity of two. CIM has no such restriction. This  
 756 causes a limitation in the mapping because CIM aggregations and compositions that have an arity larger than two  
 757 cannot be mapped to UML. Tools importing CIM-MOF into UML shall reject such CIM elements.

758 NOTE 3: The valid value combinations of the boolean CIM qualifiers *Association*, *Aggregation* and *Composition* are  
 759 not precisely defined in the [CIM Infrastructure Specification](#).

760 For the purpose of this document, the combinations that can be mapped by this document are defined in  
 761 Table 18.

762 NOTE 4: It is expected that the [CIM Infrastructure Specification](#) will define any other combination to be invalid.

763 **Table 18 – Valid Combinations for Association/Aggregation/Composition Qualifiers**

Value of <i>Association</i> Qualifier	Value of <i>Aggregation</i> Qualifier	Value of <i>Composition</i> Qualifier	Resulting Kind of CIM Class
True	False	False	Association
True	True	False	Aggregation
True	True	True	Composition
False	False	False	Normal Class

764 The *CIM::Association*, *CIM::Aggregation*, *CIM::Composition* and *CIM::Aggregate* stereotype properties of  
 765 the UML metaclass instance mapped to the qualified CIM element shall not be set.

766 **5.13.4.3 ArrayType Qualifier**

767 The *isOrdered* attribute of any UML *Property* or *Parameter* metaclass instances shall be mapped to the  
 768 effective value of the *ArrayType* qualifier on the corresponding CIM element, as defined in Table 19.

769 This mapping is motivated by the semantic equivalence of CIM Indexed with UML *isOrdered*. In both  
 770 cases, the semantic is that the array shall honor the order of array elements that is provided when setting  
 771 the array values (i.e., it does not re-order the array elements in a random or algorithmic way).

772

Table 19 – Mapping of *ArrayType* Qualifier

Value of <i>ArrayType</i> Qualifier	Value of <i>isOrdered</i> Attribute
"Bag"	False
"Ordered"	False
"Indexed"	True

773 In addition, in order to represent the distinction between *ArrayType* values "Bag" and "Ordered", the  
 774 *ArrayType* qualifier value shall be mapped generically as defined in 5.13.3, with the following additional  
 775 rules:

- 776 • The UML datatype of the *ArrayType* property of the *CIM\_Qualifier\_ArrayType* stereotype shall  
 777 be the enumeration *CIM\_Qualifier\_ArrayType\_Enum* as defined in Table 20. This enumeration  
 778 shall be owned by the UML profile "CIM".

779

Table 20 – Literals of Enumeration *CIM\_Qualifier\_ArrayType\_Enum*

Literal Name	Meaning
Bag	Represents <i>ArrayType</i> qualifier value "Bag"
Ordered	Represents <i>ArrayType</i> qualifier value "Ordered"
Indexed	Represents <i>ArrayType</i> qualifier value "Indexed"

- 780 • The value of that *ArrayType* stereotype property shall be set to the effective value of the  
 781 *ArrayType* qualifier for the particular level in the inheritance chain, as defined in Table 20.

#### 782 5.13.4.4 Deprecated Qualifier

783 The value of any *Deprecated* qualifier defined on a CIM element shall be mapped to a Marker  
 784 Stereotype as defined in 5.13.5, and in addition generically as defined in 5.13.3, with the following  
 785 additional rules:

- 786 • The *name* attribute of the Marker Stereotype shall be set to "Deprecated".
- 787 • The image of the Marker Stereotype icon should be set to a red square with the letter "D" in  
 788 white font, for example as shown in the following image:



789

- 790 • If the effective value of the CIM *Deprecated* qualifier is non-NULL on a CIM element, the Marker  
 791 Stereotype *Deprecated* shall be applied to the UML element to which that CIM element is  
 792 mapped, and the string array typed value of the *Deprecated* property of the qualifier type  
 793 stereotype *CIM\_Qualifier\_Deprecated* shall be set to the string array typed qualifier value,  
 794 preserving the order of elements.
- 795 • If the effective value of the *Deprecated* qualifier is NULL for a CIM element (i.e., it is not defined  
 796 on the element), the Marker Stereotype *Deprecated* shall not be applied to the UML element to  
 797 which that CIM element is mapped, and the string array typed value of the *Deprecated* property  
 798 of the qualifier type stereotype *CIM\_Qualifier\_Deprecated* shall be empty.

#### 799 5.13.4.5 Description Qualifier

800 The value of any *Description* qualifier defined on a CIM element shall be mapped to a *Comment*  
 801 metaclass instance owned by and associated, via *ownedComment*, with the UML metaclass instance to  
 802 which the CIM element is mapped.

803 Tool specific mappings:

- 804 • IBM Rational Software Architect:

805 The *Comment* metaclass instance shall have the *Documentation* stereotype applied.

806 Table 21 defines the mapping for the attributes and associations of the UML *Comment* metaclass.

807 **Table 21 – UML *Comment* Metaclass Used in Multiple UML Metaclasses**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
body	String	Comment	Shall be the transformed value of the CIM <i>Description</i> qualifier. The qualifier value is transformed by resolving any backslash escape sequences (for example, "\n") into their corresponding characters. See the description of the <i>string</i> datatype in the <a href="#">CIM Infrastructure Specification</a> for a definition of supported escape sequences.
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
annotatedElement	Element [1]	Comment	Shall be associated with the UML metaclass instance to which the CIM element that carries the <i>Description</i> qualifier is mapped.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.

808 The *CIM::Description* stereotype property of the UML metaclass instance mapped to the qualified CIM  
 809 element shall not be set.

810 **5.13.4.6 EmbeddedInstance Qualifier**

811 The value of any *EmbeddedInstance* qualifier defined on a CIM element shall be mapped to the *type*  
 812 association of the UML metaclass to which the CIM element is mapped. The *type* association shall  
 813 associate to the UML *Class* metaclass instance to which the CIM class specified in the qualifier value is  
 814 mapped.

815 Table 22 specifies the *CIM\_ClassReferenceType* stereotype used for UML elements mapped to CIM  
 816 properties, methods and parameters in order to specify whether instances are referenced or contained by  
 817 value.

818 **Table 22 – *CIM\_ClassReferenceType* Stereotype Used for *Property* and *Parameter* Metaclasses**

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
isAbstract	Boolean	Class	Compliance value: false (UML default).
isLeaf	Boolean	Not defined in the <a href="#">CIM Infrastructure Specification</a> at this point	If supported by the UML tool then compliance value: false.
name	String	NamedElement	Shall be "CIM_ClassReferenceType".
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private).

UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedAttribute	Property [*]	Class	Shall own and shall be associated with the stereotype properties defined in Table 23.
ownedOperation	Operation [0]	Class	Compliance value: no associated elements.
superClass	Class [0]	Class	Compliance value: no associated elements.
elementImport	ElementImport [0]	Namespace	Compliance value: no associated elements.
packageImport	PackageImport [0]	Namespace	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.
icon	Image [0]	UML <i>Stereotype</i> metaclass	Compliance value: no associated elements.

819 **Table 23 – Properties of Stereotype *CIM\_ClassReferenceType***

Name	Type	Default Value	Meaning
ClassReferenceType	Enumeration CIM_ClassReferenceType_Enum	NotApplicable	Indicates whether the instance is referenced or contained by value

820  
821 **Table 24 – Literals of Enumeration *CIM\_ClassReferenceType\_Enum***

Order Number	Literal Name	Literal Value	Meaning
1	NotApplicable	Integer 0	The element does not reference or contain an instance.
2	ByReference	Integer 1	The instance is referenced through a CIM reference.
3	ByValue	Integer 2	The instance is contained by value in an embedded instance.

822 The *CIM::EmbeddedInstance* stereotype property of the UML metaclass instance mapped to the qualified  
823 CIM element shall not be set.

824 **5.13.4.7 Experimental Qualifier**

825 The value of any *Experimental* qualifier defined on a CIM element shall be mapped to a Marker  
826 Stereotype as defined in 5.13.5, with the following additional rules:

- 827 • The *name* attribute of the Marker Stereotype shall be set to "Experimental".
- 828 • The image of the Marker Stereotype icon should be set to a blue square with the letter "E" in  
829 white font, for example as shown in the following image:



- 830
- 831 • If the effective value of the *Experimental* qualifier is *True* on a CIM element, the Marker  
832 Stereotype *Experimental* shall be applied to the UML element to which that CIM element is  
833 mapped.

- 834 • If the effective value of the *Experimental* qualifier is *False* on a CIM element, the Marker  
835 Stereotype *Experimental* shall not be applied to the UML element to which that CIM element is  
836 mapped.

837 The *CIM::Experimental* stereotype property of the UML metaclass instance mapped to the qualified CIM  
838 element shall not be set.

839 **5.13.4.8 In, Out Qualifiers**

840 The *direction* attribute of any UML *Parameter* metaclass instance shall be mapped to the effective values  
841 of the *In* and *Out* qualifiers on the corresponding CIM parameter, as defined in Table 25. All other value  
842 combinations of these qualifiers are invalid.

843 **Table 25 – Mapping of In/Out Qualifiers**

Value of <i>In</i> Qualifier	Value of <i>Out</i> Qualifier	Value of <i>direction</i> Attribute
true	false	in
false	true	out
true	true	inout

844 The *CIM::In* and *CIM::Out* stereotype properties of the UML metaclass instance mapped to the qualified  
845 CIM element shall not be set.

846 **5.13.4.9 Key Qualifier**

847 The value of any *Key* qualifier defined on a CIM element shall be mapped to a Marker Stereotype as  
848 defined in 5.13.5, with the following additional rules:

- 849 • The *name* attribute of the Marker Stereotype shall be set to "Key".
- 850 • The image of the Marker Stereotype icon should be set to a green square with the letter "K" in  
851 white font, for example as shown in the following image:



- 852 • If the effective value of the *Key* qualifier is *True* on a CIM element, the Marker Stereotype *Key*  
853 shall be applied to the UML element to which that CIM element is mapped.
- 854 • If the effective value of the *Key* qualifier is *False* on a CIM element, the Marker Stereotype *Key*  
855 shall not be applied to the UML element to which that CIM element is mapped.

857 The *CIM::Key* stereotype property of the UML metaclass instance mapped to the qualified CIM element  
858 shall not be set.

859 **5.13.4.10 Min, Max Qualifiers**

860 The *lowerValue* association end of any UML *Property* metaclass instance to which a CIM reference is  
861 mapped, shall be mapped to the effective value of the *Min* qualifier on the corresponding CIM reference.  
862 The *Property* metaclass shall own a *LiteralInteger* metaclass instance as defined in 5.16.3 whose *value*  
863 attribute is set to the *Min* qualifier value.

864 NOTE: A *Min* qualifier value of NULL cannot be mapped by this document. [DSP0004](#) clarifies that the *Min* qualifier  
865 shall not be NULL.

866 The *upperValue* association end of any UML *Property* metaclass instance to which a CIM reference is  
 867 mapped, shall be mapped to the effective value of the *Max* qualifier on the corresponding CIM reference.  
 868 The *Property* metaclass shall own a *LiteralUnlimitedNatural* metaclass instance as defined in 5.16.4  
 869 whose *value* attribute is set to the *Max* qualifier value, if non-NULL, or to "" if NULL.

870 NOTE 1: [DSP0004](#) clarifies that a NULL value for the *Max* qualifier means an upper value of unlimited for the  
 871 cardinality of the so qualified CIM reference.

872 NOTE 2: UML uses the multiplicity elements *upperValue* and *lowerValue* in the *Property* metaclass slightly different,  
 873 depending on whether or not the *Property* is used as an association end of an association that owns its association  
 874 ends. If the association ends are owned by the association, the multiplicity elements still indicate the multiplicity of the  
 875 associated classes, but that is different from the multiplicity of the owned association ends themselves (which is  
 876 always "1"). The mapping of the *Min* and *Max* qualifiers follows the UML definition of multiplicity for association ends  
 877 owned by the association.

878 The *CIM::Min* and *CIM::Max* stereotype properties of the UML metaclass instance mapped to the qualified  
 879 CIM element shall not be set.

#### 880 5.13.4.11 OCL Constraint Related Qualifiers

881 The [CIM Infrastructure Specification](#) defines a qualifier named *OCL*. [DSP0004](#) deprecates the *OCL*  
 882 qualifier and adds three new OCL constraint related qualifiers (*ClassConstraint*, *MethodConstraint* and  
 883 *PropertyConstraint*). This document maps these three new qualifiers and ignores the deprecated *OCL*  
 884 qualifier.

885 Each array entry in any CIM *ClassConstraint*, *MethodConstraint* and *PropertyConstraint* qualifier value  
 886 defined on a CIM element shall be mapped to a UML *Constraint* metaclass instance as defined in  
 887 Table 27. That *Constraint* metaclass instance shall be associated with the UML metaclass instance to  
 888 which the CIM element defining the qualifier is mapped, via the UML associations defined in Table 26.

889 **Table 26 – Mapping of OCL Constraint Related Qualifiers**

CIM Qualifier	CIM Elements the Qualifier is Used On	OCL Constraint Type	Constraint::constrainedElement	Constraint::context
ClassConstraint	Class Association Class Indication	def:	UML <i>Class</i> instance to which the qualified CIM class is mapped.	UML <i>Class</i> instance to which the qualified CIM class is mapped.
ClassConstraint	Class Association Class Indication	inv:	UML <i>Class</i> instance to which the qualified CIM class is mapped.	UML <i>Class</i> instance to which the qualified CIM class is mapped.
MethodConstraint	Method	body:	UML <i>Operation</i> instance to which the qualified CIM method is mapped.	UML <i>Class</i> instance owning the UML <i>Operation</i> instance to which the qualified CIM method is mapped.
MethodConstraint	Method	pre:	UML <i>Operation</i> instance to which the qualified CIM method is mapped.	UML <i>Class</i> instance owning the UML <i>Operation</i> instance to which the qualified CIM method is mapped.
MethodConstraint	Method	post:	UML <i>Operation</i> instance to which the qualified CIM method is mapped.	UML <i>Class</i> instance owning the UML <i>Operation</i> instance to which the qualified CIM method is mapped.



CIM Qualifier	CIM Elements the Qualifier is Used On	OCL Constraint Type	Constraint::constrainedElement	Constraint::context
PropertyConstraint	Property Reference	init:	UML <i>Class</i> instance owning the UML <i>Property</i> instance to which the qualified CIM property is mapped.	UML <i>Class</i> instance owning the UML <i>Property</i> instance to which the qualified CIM property is mapped.
PropertyConstraint	Property Reference	derive:	UML <i>Class</i> instance owning the UML <i>Property</i> instance to which the qualified CIM property is mapped.	UML <i>Class</i> instance owning the UML <i>Property</i> instance to which the qualified CIM property is mapped.

890 **Table 27 – Usage of UML *Constraint* Metaclass for OCL Related Constraint Qualifiers**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
visibility	VisibilityKind	PackagableElement	No compliance value is defined.  NOTE: It is not clear what visibility should mean on a constraint, OMG issue #10382 has been raised to clarify that, targeting UML 2.2.
name	String [0]	NamedElement	No compliance value is defined; extension point.
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
constrainedElement	Element [1]	Constraint	Shall be associated with the UML metaclass instance specified in the <i>Constraint::constrainedElement</i> column in Table 26.
context	Namespace [1]	Constraint	Shall be associated with the UML metaclass instance specified in the <i>Constraint::context</i> column in Table 26. <sup>1</sup>
specification	ValueSpecification	Constraint	Shall own and shall be associated with an <i>OpaqueExpression</i> metaclass instance as defined in Table 28.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.

891  
892 <sup>1</sup> In the [UML Superstructure Specification V2.1.1](#), the *context* association is a derived association. An OMG issue has been  
893 raised proposing that *context* gets changed to a non-derived association. This document assumes *context* is not derived and  
894 therefore defines its mapping.

895  
896**Table 28 – Usage of UML *OpaqueExpression* Metaclass Used for OCL Related Constraint Qualifiers**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
visibility	VisibilityKind	PackagableElement	No compliance value defined.
name	String [0]	NamedElement	Compliance value: no value.
body	String [1]	OpaqueExpression	Shall be the value of the array entry of the OCL related constraint qualifier.
language	String [1]	OpaqueExpression	Shall be the string "OCL".
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
type	Type	TypedElement	Shall be associated with the <i>PrimitiveType</i> metaclass instance representing the datatype UML::Boolean.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.

897 The *CIM::ClassConstraint*, *CIM::MethodConstraint* and *CIM::PropertyConstraint* stereotype properties of  
898 the UML metaclass instance mapped to the qualified CIM element shall not be set.

#### 899 5.13.4.12 Octetstring Qualifier

900 The datatype of any CIM element on which the CIM *Octetstring* qualifier has an effective value of *true*  
901 shall be mapped as defined in 5.11.

902 The values of CIM elements qualified with *Octetstring* that have a CIM datatype array of uint8 shall be  
903 transformed into a hexadecimal string format when represented in the UML element to which the CIM  
904 element is mapped. The hexadecimal string format shall conform to the format defined in the [CIM](#)  
905 [Infrastructure Specification](#) for octetstring values of type array of string.

906 The values of CIM elements qualified with *Octetstring* that have a CIM datatype array of string shall be  
907 used unchanged when represented in the UML element to which the CIM element is mapped.

908 The *CIM::Octetstring* stereotype property of the UML metaclass instance mapped to the qualified CIM  
909 element shall not be set.

#### 910 5.13.4.13 Override Qualifier

911 The value of any *Override* qualifier defined on a CIM element shall be mapped as follows:

- 912 • For CIM properties and references, to the *redefinedProperty* association of the UML *Property*  
913 metaclass to which the CIM property or reference carrying the *Override* qualifier is mapped. The  
914 *redefinedProperty* association shall reference the UML *Property* metaclass instance to which  
915 the CIM property or reference being overridden is mapped.
- 916 • For CIM methods, to the *redefinedOperation* association of the UML *Operation* metaclass to  
917 which the CIM method carrying the *Override* qualifier is mapped. The *redefinedOperation*  
918 association shall reference the UML *Operation* metaclass instance to which the CIM property or  
919 reference being overridden is mapped.

920 NOTE: The class of the CIM element being overridden is either specified in the *Override* qualifier value, or is the next  
921 class defining the element when searching towards the superclasses from the CIM class at which level the *Override*  
922 qualifier is defined.

923 The *CIM::Override* stereotype property of the UML metaclass instance mapped to the qualified CIM  
 924 element shall not be set.

#### 925 5.13.4.14 Static Qualifier

926 The *isStatic* attribute of any UML *Property* and *Operation* metaclass instances shall be equal to the  
 927 effective value of the *Static* qualifier on the corresponding CIM element.

928 The *CIM::Static* stereotype property of the UML metaclass instance mapped to the qualified CIM element  
 929 shall not be set.

#### 930 5.13.4.15 Terminal Qualifier

931 The *isLeaf* attribute of any UML *Class* metaclass instance shall be equal to the effective value of the  
 932 *Terminal* qualifier on the corresponding CIM element.

933 The *CIM::Terminal* stereotype property of the UML metaclass instance mapped to the qualified CIM  
 934 element shall not be set.

#### 935 5.13.4.16 Values and ValueMap Qualifiers

936 If the effective values of the *Values* qualifier or the *ValueMap* qualifier or both are non-NULL, the qualified  
 937 CIM element shall be mapped using the "*ValueMap mapping*" defined in this subclause.

938 The UML profile "CIM" shall define and own:

- 939 • A UML stereotype *CIM\_Enumeration* as defined in Table 30.

940 In the user model, the qualifier values of *Values* and *ValueMap* shall be mapped to UML enumerations as  
 941 follows:

- 942 • A UML *Enumeration* metaclass instance as defined in Table 29 shall be owned by and defined  
 943 in the scope of the UML class to which the CIM class owning the qualified CIM element is  
 944 mapped.
- 945 • The stereotype *CIM\_Enumeration* defined in Table 30 shall be applied to that UML *Enumeration*  
 946 metaclass instance.
- 947 • The *type* association of the UML metaclasses *Property* and *Parameter* to which the CIM  
 948 element qualified with *Values* or *ValueMap* is mapped shall be associated with that UML  
 949 *Enumeration* metaclass instance.
- 950 • For each pair of entries in the *Values* or *ValueMap* arrays, a UML *EnumerationLiteral* metaclass  
 951 instance as defined in Table 32 shall be owned by that UML *Enumeration* metaclass instance.

952 The UML enumeration is defined in the scope of the CIM class that owns the qualified CIM element in  
 953 order to support the downward compatible changes to the CIM Schema defined in the [CIM Infrastructure  
 954 Specification](#) without changing the name of the enumeration (e.g., moving the qualified element into a  
 955 superclass).

956 The representation of *Values* and *ValueMap* qualifier values as UML *EnumerationLiteral* metaclass  
 957 instances always has both name and value specification, regardless of whether only one of the *Values*  
 958 and *ValueMap* qualifiers was defined when importing from CIM MOF. The stereotype properties  
 959 *IsValuesDefined* and *IsValueMapDefined* allow supporting round-trip engineering between CIM MOF  
 960 format and UML model. They may be used during export of a UML model to CIM MOF format for deciding  
 961 whether the *Values* and *ValueMap* qualifiers need to be generated.

962 CIM allows the introduction of a ValueMap qualifier on an overriding element (i.e., if there was no  
 963 valueMap qualifier defined on the overridden element). This is supported in a type compatible way by the  
 964 mapping to UML such that the UML *Enumeration* metaclass instance used as a UML datatype for the  
 965 overriding element is a specialization of the CIM datatype used for the overridden element.

966 **Table 29 – UML Enumeration Metaclass Used for Values and ValueMap Qualifiers**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
isAbstract	Boolean	Classifier	Compliance value: false (UML default).
isLeaf	Boolean	RedefinableElement	Compliance value: false (UML default).
name	String	NamedElement	If the qualified CIM element is a property or a method, <i>name</i> shall be the name of the element in unchanged lexical case, followed by the string "_Enum".  If the qualified CIM element is a parameter, <i>name</i> shall be the name of the method in unchanged lexical case, followed by the string "_", followed by the name of the parameter in unchanged lexical case, followed by the string "_Enum".
visibility	VisibilityKind	PackagableElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
ownedLiteral	EnumerationLiteral [*]	Enumeration	Shall own and shall be associated with an <i>EnumerationLiteral</i> metaclass instance for each pair of <i>Values</i> and <i>ValueMap</i> qualifier values. The order of pairs shall be preserved.
ownedAttribute	Property [0]	DataType	Compliance value: no associated elements.
ownedOperation	Operation [0]	DataType	Compliance value: no associated elements.
generalization	Generalization [1]	Classifier	Shall own and shall be associated with a <i>Generalization</i> metaclass instance as defined in 5.16.6 whose <i>general</i> association shall be associated with a <i>Classifier</i> as follows: If the CIM element qualified with <i>ValueMap</i> is overriding another element that has an effective <i>ValueMap</i> value other than NULL, with the <i>Enumeration</i> instance to which that overridden <i>ValueMap</i> value is mapped. Otherwise, with an instance of the <i>DataType</i> metaclass instance to which the CIM datatype of the element qualified with <i>ValueMap</i> is mapped, as defined in Table 8. In other words, the first Enumeration in an override chain specializes the CIM datatype, and any further Enumerations specialize the previous Enumeration.
redefinedClassifier	Classifier [0]	Classifier	Compliance value: no associated elements.

967

UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
package	Package [0]	Type (since UML 2.1)	Compliance value: no associated elements.  Note that the reason for this is that the <i>Enumeration</i> is owned by a <i>Class</i> , not by a <i>Package</i> .  Note that the derived association end <i>owner</i> allows traversing to the <i>Class</i> owning this <i>Enumeration</i> .
elementImport	ElementImport [0]	Namespace	Compliance value: no associated elements.
ownedRule	Constraint [*]	Namespace	No compliance value is defined; extension point.
packageImport	PackageImport [0]	Namespace	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.
Stereotype Property	UML Type	Defined in Stereotype	CIM Mapping
isValuesDefined	Boolean	CIM_Enumeration	Shall be <i>True</i> (default) if the <i>Values</i> qualifier is to be defined on the qualified CIM element in its MOF representation, and <i>False</i> otherwise.  If set to <i>False</i> , the <i>name</i> attribute of each enumeration literal of this enumeration shall match the value associated with the <i>specification</i> association.
isValueMapDefined	Boolean	CIM_Enumeration	Shall be <i>True</i> (default) if the <i>ValueMap</i> qualifier is defined on the qualified CIM element, and <i>False</i> otherwise.  If set to <i>False</i> , the value associated with the <i>specification</i> association of each enumeration literal shall be an ordered list starting with 0 and incrementing by 1.

968  
969

**Table 30 – CIM\_Enumeration Stereotype Used for Enumeration Metaclass Used for Values and ValueMap Qualifiers**

UML attribute	UML Infrastructure Library type	Defined in UML Infrastructure Library metaclass	CIM Mapping
isAbstract	Boolean	Class	Compliance value: false (UML default).
isLeaf	Boolean	Not defined in <a href="#">CIM Infrastructure Specification</a> at this point	If supported by the UML tool then compliance value: false.
name	String	NamedElement	Shall be "CIM_Enumeration".
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private).

UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedAttribute	Property [*]	Class	Shall own and shall be associated with the stereotype properties defined in Table 31.
ownedOperation	Operation [0]	Class	Compliance value: no associated elements.
superClass	Class [0]	Class	Compliance value: no associated elements.
elementImport	ElementImport [0]	Namespace	Compliance value: no associated elements.
packageImport	PackageImport [0]	Namespace	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.
icon	Image [0]	UML <i>Stereotype</i> metaclass	Compliance value: no associated elements.

970

Table 31 – Properties of Stereotype *CIM\_Enumeration*

Name	Type	Default Value	Meaning
IsValuesDefined	Boolean	True	Indicates that the <i>Values</i> qualifier is defined on the CIM element.
IsValueMapDefined	Boolean	True	Indicates that the <i>ValueMap</i> qualifier is defined on the CIM element.

971

Table 32 – UML *EnumerationLiteral* Metaclass Used for Pair of *Values* and *ValueMap* Qualifier Values

972

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
name	String	NamedElement	Shall be the value of the <i>Values</i> qualifier of the pair, if a <i>Values</i> qualifier is defined. Otherwise, shall be the value of the <i>ValueMap</i> qualifier of the pair. In any case, the backslash character (\) as well as any characters or character sequences invalid for UML names (for example, double colons) need to be escaped with a backslash.
visibility	VisibilityKind	PackagableElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
enumeration	Enumeration [1]	EnumerationLiteral	Shall be associated with the UML Enumeration metaclass instance owning this EnumerationLiteral metaclass instance.
classifier	Classifier [1]	InstanceSpecification	No compliance value defined. NOTE: OMG issue #9841 suggests two alternatives on how to constrain this association.
slot	Slot [0]	InstanceSpecification	Compliance value: no associated elements. NOTE: This is not defined in the <a href="#">UML Superstructure Specification</a> but suggested in OMG issue #9841.

973

UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
specification	ValueSpecification [1]	InstanceSpecification	<p>Shall own and shall be associated with an instance of an appropriate subclass of the UML <i>ValueSpecification</i> metaclass whose <i>value</i> attribute shall be set as follows:</p> <p>To the value of the <i>ValueMap</i> qualifier of the pair, if a <i>ValueMap</i> qualifier is defined. Non-range values should be represented as integers, and range values as strings.</p> <p>Otherwise, to an integer value reflecting the relative order of the entry within the <i>Values</i> array, starting with 0. Note this is the default rule defined in <a href="#">DSP004</a>.</p> <p>Note that the usage of the <i>specification</i> association for the value of the enumeration literal is not defined in the <a href="#">UML Superstructure Specification</a> but suggested in OMG issue #9842.</p>
ownedComment	Comment [0..1]	Element	No compliance value is defined; extension point.

974 The *CIM::ValueMap* and *CIM::Values* stereotype properties of the UML metaclass instance mapped to  
 975 the qualified CIM element shall not be set.

976 **5.13.4.17 Write Qualifier**

977 The *isReadOnly* attribute of any UML *Property* metaclass instance shall be mapped to the effective value  
 978 of the *Write* qualifier on the corresponding CIM element, as defined in Table 33.

979 **Table 33 – Mapping of *Write* Qualifier**

Value of <i>Write</i> Qualifier	Value of <i>isReadOnly</i> Attribute
True	False
False	True

980 The *CIM::Write* stereotype property of the UML metaclass instance mapped to the qualified CIM element  
 981 shall not be set.

982 **5.13.5 Definition of Marker Stereotypes**

983 As defined in 5.13.4, some directly mapped qualifier types use Marker Stereotypes. This subclause  
 984 defines the Marker Stereotypes for this purpose.

985 Table 34 defines the mapping for the attributes and associations of the UML *Stereotype* metaclass for  
 986 use as a Marker Stereotype.

987 The UML *Stereotype* metaclass has mostly attributes and associations from the UML Infrastructure  
 988 Library defined in the [UML Infrastructure Specification](#).

989

Table 34 – UML *Stereotype* Metaclass Used as Marker Stereotype

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
isAbstract	Boolean	Class	Compliance value: false (UML default).
isLeaf	Boolean	Not defined in the <a href="#">UML Infrastructure Specification</a> at this point	If supported by the UML tool then compliance value: false.
name	String	NamedElement	The mapping of this attribute is defined in the description of the qualifier that is using this marker stereotype, in 5.13.4.
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedAttribute	Property [0..1]	Class	Depends on the usage context for the marker stereotype. The rules are defined in each usage.
ownedOperation	Operation [0]	Class	Compliance value: no associated elements.
superClass	Class [0]	Class	Compliance value: no associated elements.
elementImport	ElementImport [0]	Namespace	Compliance value: no associated elements.
packageImport	PackageImport [0]	Namespace	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.
icon	Image [1]	UML <i>Stereotype</i> metaclass	Shall own and shall be associated with an <i>Image</i> metaclass instance as defined in Table 35.

990

Table 35 – UML *Image* Metaclass Used by Marker Stereotype

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
content	String [1]	UML <i>Image</i> metaclass	Shall contain the serialization of the image of the stereotype icon, in the format defined by the format attribute. The image of the stereotype icon is defined in the description of the qualifier that is using this marker stereotype, in 5.13.4.
format	String [1]	UML <i>Image</i> metaclass	Shall contain the format used for the content attribute.
location	String [0]	UML <i>Image</i> metaclass	Compliance value: no elements.
UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedComment	Comment [0]	Element	Compliance value: no associated elements.



### 991 5.13.6 Typical Use Cases for Qualifier Inheritance Mapping

992 This subclause provides algorithms for some typical use cases that are derived from the mapping rules  
993 defined in the previous subclause.

#### 994 5.13.6.1 Use Case: Importing CIM MOF into a UML Tool

995 Determination of the CIM qualifier value to be used, and whether the UML construct to which the qualifier  
996 is mapped, needs to be created:

- 997 • For UML requirements level = Required:

998 The UML construct is already present. Set its value from the effective qualifier value in the CIM  
999 MOF.

- 1000 • For UML requirements level = Optional:

1001 If a qualifier value is defined in the CIM MOF with a value other than the default value (for  
1002 *Restricted* qualifiers) or the inherited value (for *ToSubclass* qualifiers):

- 1003 – then create the UML construct and set its value from the qualifier value defined in the CIM  
1004 MOF;
- 1005 – else do not create the UML construct.

1006 NOTE: Determining the inherited value or the effective qualifier value requires knowledge about the superclasses, so  
1007 these superclasses need to be imported first. This results in automatically propagating the values down the class  
1008 hierarchy towards the leaf nodes as classes are imported.

1009 For qualifiers with *DisableOverride*, the import logic may validate that the *DisableOverride* constraint is  
1010 not violated.

#### 1011 5.13.6.2 Use Case: Editing a UML User Model with a UML Tool

1012 Displaying the effective qualifier value at a given class level:

- 1013 • If the corresponding UML construct exists in the class, then determine the qualifier value from  
1014 that construct.
- 1015 • Else, use the default qualifier value (for *Restricted* qualifiers) or the inherited value (for  
1016 *ToSubclass* qualifiers).

1017 NOTE: It is useful to also display the origin of the effective qualifier value along with the value.

1018 Modification of a qualifier value at a particular class level:

- 1019 • For UML requirements level = Required:

1020 The UML construct is already present. Set its value from the new qualifier value.

- 1021 • For UML requirements level = Optional:

1022 If the new qualifier value is the default value (for *Restricted* qualifiers) or the inherited value (for  
1023 *ToSubclass* qualifiers):

- 1024 – then remove the UML construct if it exists;
- 1025 – else create the UML construct if it does not exist and set its value from the new qualifier  
1026 value.

1027 **5.13.6.3 Use Case: Exporting from a UML Tool into CIM MOF**

1028 Determining whether a qualifier value definition should be created in the exported CIM MOF:

- 1029
- 1030
- 1031
- If the CIM qualifier is in the set (Association, Aggregation, Composition, Indication, Exception, In) and its effective value is *True*, then create a qualifier definition in the exported CIM MOF. Note that this is needed for compatibility with existing CIM tools.
  - Else, if the corresponding UML construct exists in the class that is exported, AND if the qualifier value is not equal to the default value (for *Restricted* qualifiers) or the inherited value (for *ToSubclass* qualifiers):
    - then create a qualifier definition in the exported CIM MOF;
    - else do not create a qualifier definition in the exported CIM MOF.

1037 **5.14 Mapping MOF File Information**

1038 CIM MOF files contain information in addition to CIM classes and instances. In order to support

1039 roundtripping from CIM MOF to UML and back, this information needs to be represented in UML. This

1040 subclause normatively defines how information related to CIM MOF files is mapped to UML.

1041 The following information related to CIM MOF files is mapped to UML:

- 1042
- 1043
- 1044
- 1045
- 1046
- 1047
- The name of the CIM MOF file
  - The directory path of the CIM MOF file
  - The ordered list of header lines in the CIM MOF file
  - The value of the `#pragma locale` directive
  - The value of the `#pragma instancelocale` directive
  - The value of the `#pragma namespace` directive

1048 The ordered list of header lines is defined to be the lines at the top of the MOF file before the first non-

1049 empty and non-comment line.

1050 CIM MOF files have no directly corresponding entity in the UML user model. Therefore, the information

1051 items listed above are mapped to properties of stereotypes applied to the classes, instances and

1052 packages in the UML user model, as defined in Table 36:

1053 **Table 36 – Mapping of CIM MOF File Related Information**

CIM MOF File Related Information Item	Stereotype Property	Defined in Stereotype	Applied to UML Metaclass
Name of the CIM MOF file	FileName	CIM_MofFileInfo	Class (required) InstanceSpecification (required) Package (required)
Directory path of the CIM MOF file	DirectoryPath		
Ordered list of header lines in the CIM MOF file	HeaderLines		
Value of the <code>#pragma locale</code> directive	ClassLocale	CIM_ClassLocale	Class (required) Package (required)

CIM MOF File Related Information Item	Stereotype Property	Defined in Stereotype	Applied to UML Metaclass
Value of the <code>#pragma instancelocale</code> directive	InstanceLocale	CIM_InstanceLocale	InstanceSpecification (required) Package (required)
Value of the <code>#pragma namespace</code> directive	NamespacePath	CIM_NamespacePath	Class (required) InstanceSpecification (required) Package (required)

1054 These stereotypes and their properties are defined in 5.14.1.

1055 As defined in Table 36, each such information item may be set at a package level in addition to the class  
 1056 or instance level. The value of each such information item in effect for a CIM class or CIM instance shall  
 1057 be determined as follows:

- 1058 1) If the information item has a non-null value at the class or instance, that value is used.
- 1059 2) Else, if the information item has a non-null value at the package containing the class or  
 1060 instance, that value is used.
- 1061 3) Else, the packages are searched outwards until a non-null value is found.
- 1062 4) If no non-null value is found at the outermost package of the UML user model, the effective  
 1063 value is null.

1064 This algorithm is used for each information item separately, so the values in effect for a class or instance  
 1065 may be defined at different levels.

1066 The information items are represented from the point of view of a CIM class or CIM instance. This does  
 1067 not allow reconstructing a CIM MOF file exactly as it was before importing it into UML. However, it allows  
 1068 reconstructing a CIM MOF file that has equivalent information as the imported CIM MOF file. For  
 1069 example, two consecutive `#pragma locale` directives before a CIM class definition in the original CIM  
 1070 MOF file get mapped to a single *ClassLocale* property of that class in the UML user model, and result in a  
 1071 single `#pragma locale` directive when exported to CIM MOF.

1072 **5.14.1 Stereotypes Used for Mapping MOF File Information**

1073 **5.14.1.1 Stereotype *CIM\_ClassLocale***

1074 The stereotype *CIM\_ClassLocale* represents a `#pragma locale` directive in the CIM MOF file. This  
 1075 stereotype shall be defined in the UML profile "CIM", as defined in Table 37:

1076 **Table 37 – Stereotype *CIM\_ClassLocale***

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
isAbstract	Boolean	Class	Compliance value: false.
name	String	NamedElement	Shall be the string "CIM_ClassLocale".
Other attributes as defined in Table 47.			

UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedAttribute	Property	Class	Shall be associated with and own the property defined in Table 38.
Other associations as defined in Table 47.			

1077

**Table 38 – Property *ClassLocale* Owned by Stereotype *CIM\_ClassLocale***

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
name	String	NamedElement	Shall be the string "ClassLocale".
Other attributes as defined in Table 48.			
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
defaultValue	ValueSpecification	Property	Shall own and shall be associated with an instance of the <i>LiteralNull</i> metaclass.
type	Type	TypedElement	Shall be associated with the <i>PrimitiveType</i> metaclass instance representing the datatype UML::String.
lowerValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
upperValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
Other associations as defined in Table 48.			

1078

**5.14.1.2 Stereotype *CIM\_InstanceLocale***

1079

The stereotype *CIM\_InstanceLocale* represents a #pragma *instancelocale* directive in the CIM MOF file. This stereotype shall be defined in the UML profile "CIM", as defined in Table 39:

1080

1081

**Table 39 – Stereotype *CIM\_InstanceLocale***

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
isAbstract	Boolean	Class	Compliance value: false.
name	String	NamedElement	Shall be the string "CIM_InstanceLocale".
Other attributes as defined in Table 47.			
UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedAttribute	Property	Class	Shall be associated with and own the property defined in Table 40.
Other associations as defined in Table 47.			

1082

**Table 40 – Property *InstanceLocale* Owned by Stereotype *CIM\_InstanceLocale***

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
name	String	NamedElement	Shall be the string "InstanceLocale".
Other attributes as defined in Table 48			
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
defaultValue	ValueSpecification	Property	Shall own and shall be associated with an instance of the <i>LiteralNull</i> metaclass.
type	Type	TypedElement	Shall be associated with the <i>PrimitiveType</i> metaclass instance representing the datatype UML::String.
lowerValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
upperValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
Other associations as defined in Table 48.			

1083 **5.14.1.3 Stereotype *CIM\_NamespacePath***

1084 The stereotype *CIM\_NamespacePath* represents a #pragma namespace directive in the CIM MOF file.

1085 This stereotype shall be defined in the UML profile "CIM", as defined in Table 41:

1086

**Table 41 – Stereotype *CIM\_NamespacePath***

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
isAbstract	Boolean	Class	Compliance value: false.
name	String	NamedElement	Shall be the string "CIM_NamespacePath".
Other attributes as defined in Table 47.			
UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedAttribute	Property	Class	Shall be associated with and own the property defined in Table 42.
Other associations as defined in Table 47.			

1087

Table 42 – Property *NamespacePath* Owned by Stereotype *CIM\_NamespacePath*

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
name	String	NamedElement	Shall be the string "NamespacePath".
Other attributes as defined in Table 48.			
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
defaultValue	ValueSpecification	Property	Shall own and shall be associated with an instance of the <i>LiteralNull</i> metaclass.
type	Type	TypedElement	Shall be associated with the <i>PrimitiveType</i> metaclass instance representing the datatype UML::String.
lowerValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
upperValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
Other associations as defined in Table 48.			

1088 **5.14.1.4 Stereotype *CIM\_MofFileInfo***

1089 The stereotype *CIM\_MofFileInfo* represents information related to a CIM MOF file. This stereotype shall  
 1090 be defined in the UML profile "CIM", as defined in Table 43:

1091

Table 43 – Stereotype *CIM\_MofFileInfo*

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
isAbstract	Boolean	Class	Compliance value: false.
name	String	NamedElement	Shall be the string "CIM_MofFileInfo".
Other attributes as defined in Table 47.			
UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedAttribute	Property [3]	Class	Shall be associated with and own the properties defined in Table 44, Table 45 and Table 46.
Other associations as defined in Table 47.			

1092

**Table 44 – Property *FileName* Owned by Stereotype *CIM\_MofFileInfo***

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
name	String	NamedElement	Shall be the string "FileName".
Other attributes as defined in Table 48.			
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
defaultValue	ValueSpecification	Property	Shall own and shall be associated with an instance of the <i>LiteralNull</i> metaclass.
type	Type	TypedElement	Shall be associated with the <i>PrimitiveType</i> metaclass instance representing the datatype UML::String.
lowerValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
upperValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
Other associations as defined in Table 48.			

1093

**Table 45 – Property *DirectoryPath* Owned by Stereotype *CIM\_MofFileInfo***

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
name	String	NamedElement	Shall be the string "DirectoryPath".
Other attributes as defined in Table 48.			
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
defaultValue	ValueSpecification	Property	Shall own and shall be associated with an instance of the <i>LiteralNull</i> metaclass.
type	Type	TypedElement	Shall be associated with the <i>PrimitiveType</i> metaclass instance representing the datatype UML::String.
lowerValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
upperValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
Other associations as defined in Table 48.			

1094

Table 46 – Property *HeaderLines* Owned by Stereotype *CIM\_MofFileInfo*

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
name	String	NamedElement	Shall be the string "HeaderLines".
Other attributes as defined in Table 48.			
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
defaultValue	ValueSpecification	Property	Shall own and shall be associated with an instance of the <i>Literal/Null</i> metaclass.
type	Type	TypedElement	Shall be associated with the <i>PrimitiveType</i> metaclass instance representing the datatype UML::String.
lowerValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
upperValue	ValueSpecification	MultiplicityElement	Shall have the value "1" in a UML2 defined representation.
Other associations as defined in Table 48.			

1095 **5.14.1.5 Other Definitions for Stereotypes Used for Mapping MOF File Information**

1096 The stereotypes and their properties used for mapping MOF file information shall be compliant with the  
 1097 additional definitions for their attributes and associations, as defined in Table 47 and Table 48.

1098

Table 47 – Other Definitions for Stereotypes Used for Mapping MOF File Information

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
isAbstract	Boolean	Class	Compliance value: false.
isLeaf	Boolean	Not defined in the <a href="#">UML Infrastructure Specification</a> at this point	If supported by the UML tool then compliance value: false.
name	String	NamedElement	As defined in the table referring to this table.
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedAttribute	Property	Class	As defined in the table referring to this table.
ownedOperation	Operation [0]	Class	Compliance value: no associated elements.
superClass	Class [0]	Class	Compliance value: no associated elements.
elementImport	ElementImport [0]	Namespace	Compliance value: no associated elements.
packageImport	PackageImport [0]	Namespace	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.
icon	Image [0]	UML <i>Stereotype</i> metaclass	Compliance value: no associated elements.



1099 **Table 48 – Other Definitions for Properties of Stereotypes Used for Mapping MOF File Information**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
aggregation	AggregationKind	Property	Compliance value: none (UML default).
isDerived	Boolean	Property	Compliance value: false (UML default).
isDerivedUnion	Boolean	Property	Compliance value: false (UML default).
isReadOnly	Boolean	Property	Compliance value: false (UML default).
isStatic	Boolean	Feature	Compliance value: false (UML default).
isLeaf	Boolean	RedefinableElement	Compliance value: false (UML default).
name	String	NamedElement	As defined in the table referring to this table.
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private).
isOrdered	Boolean	MultiplicityElement	Compliance value: false (UML default).
isUnique	Boolean	MultiplicityElement	Compliance value: false (UML default is true).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
association	Association [0]	Property	Compliance value: no associated elements.
owningAssociation	Association [0]	Property	Compliance value: no associated elements.
datatype	DataType [0]	Property	Compliance value: no associated elements (because this property is not owned by a structured datatype).
defaultValue	ValueSpecification	Property	As defined in the table referring to this table.
redefinedProperty	Property [0]	Property	Compliance value: no associated elements.
subsettedProperty	Property [0]	Property	Compliance value: no associated elements.
associationEnd	Property [0]	Property	Compliance value: no associated elements.
qualifier	Property [0]	Property	Compliance value: no associated elements.
type	Type	TypedElement	As defined in the table referring to this table.
lowerValue	ValueSpecification	MultiplicityElement	As defined in the table referring to this table.
upperValue	ValueSpecification	MultiplicityElement	As defined in the table referring to this table.
ownedComment	Comment [*]	Element	Compliance value: no associated elements.
class (since UML 2.1)	Class	Property	Shall be owned by and associated with the <i>Stereotype</i> metaclass instance owning the property.

1100 **5.15 Stereotypes for the CIM Meta Elements**

1101 This subclause normatively defines those stereotypes that may be applied to the CIM meta elements.

1102 These stereotypes are called "Meta Element Stereotypes" in this document.

1103 NOTE: Some of them inherit from the Qualifier Scope Stereotypes. This is how the CIM meta elements get the ability  
1104 to have CIM qualifier values.

1105 The UML profile "CIM" shall define and own the stereotypes defined in Table 49. As an extension point,  
1106 these stereotypes may define additional properties, and may inherit from additional base stereotypes.

1107

Table 49 – Meta Element Stereotypes

Stereotype Name	Abstract	Extends UML Metaclass	Base Stereotypes and Properties
CIM_Package	No	Package (Required)	No base stereotypes required
CIM_Schema	No	Package (Optional)	No base stereotypes required
CIM_Class	No	Class (Optional)	CIM_Class_Qualifiers
CIM_Association	No	AssociationClass (Optional)	CIM_Association_Qualifiers
CIM_Indication	No	Class (Optional)	CIM_Indication_Qualifiers
CIM_Property	No	Property (Optional)	CIM_Property_Qualifiers CIM_ClassReferenceType
CIM_Reference	No	Property (Optional)	CIM_Reference_Qualifiers
CIM_Method	No	Operation (Required)	CIM_Method_Qualifiers
CIM_Parameter	No	Parameter (Optional)	CIM_Parameter_Qualifiers CIM_ClassReferenceType
CIM_Instance	No	InstanceSpecification (Optional)	No base stereotypes required
CIM_Enumeration	No	Enumeration (Optional)	No base stereotypes required

1108 Any other attributes or associations of the Meta Element Stereotypes shall be as defined in Table 50.

1109

Table 50 – Definition of Meta Element Stereotypes

UML Attribute	UML Infrastructure Library Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
isAbstract	Boolean	Class	As defined in the table referring to this table.
isLeaf	Boolean	Not defined in the <a href="#">UML Infrastructure Specification</a> at this point	If supported by the UML tool then compliance value: false.
name	String	NamedElement	As defined in the table referring to this table.
visibility	VisibilityKind	NamedElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Infrastructure Library Metaclass	CIM Mapping
ownedAttribute	Property [*]	Class	As defined in the table referring to this table.
ownedOperation	Operation [0]	Class	Compliance value: no associated elements.
superClass	Class [*]	Class	As defined in the table referring to this table.
elementImport	ElementImport [0]	Namespace	Compliance value: no associated elements.
packageImport	PackageImport [0]	Namespace	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.
icon	Image [*]	UML <i>Stereotype</i> metaclass	No compliance value is defined; extension point.

1110 **5.16 Other UML Metaclasses Used in the Mapping**

1111 This subclause normatively defines the usage of other UML metaclasses in addition to those defined in  
 1112 the previous subclauses.

1113 **5.16.1 UML *LiteralBoolean* Metaclass**

1114 The UML *LiteralBoolean* metaclass as defined in this subclause is used to represent boolean values in  
 1115 the following UML metaclasses:

- 1116 • *defaultValue* association in UML *Property* metaclass used for CIM properties, as defined in  
 1117 Table 4.
- 1118 • *value* association in UML *Slot* metaclass used for property values of modeled CIM instances, as  
 1119 defined in Table 10.
- 1120 • *defaultValue* association in UML *Property* metaclass used for CIM qualifiers in Qualifier Type  
 1121 Stereotypes, as defined in Table 12.

1122 Table 51 defines the mapping for the attributes and associations of the UML *LiteralBoolean* metaclass for  
 1123 these usages.

1124 **Table 51 – UML *LiteralBoolean* Metaclass**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
value	Boolean	LiteralBoolean	Shall be the boolean value to be represented.
name	String [0]	NamedElement	Compliance value: no value.
visibility	VisibilityKind	PackagableElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
type	Type [0]	TypedElement	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.

1125 **5.16.2 UML *LiteralString* Metaclass**

1126 The UML *LiteralString* metaclass as defined in this subclause is used to represent string values in the  
 1127 following UML metaclasses:

- 1128 • *defaultValue* association in UML *Property* metaclass used for CIM properties, as defined in  
 1129 Table 4.
- 1130 • *value* association in UML *Slot* metaclass used for property values of modeled CIM instances, as  
 1131 defined in Table 10.
- 1132 • *defaultValue* association in UML *Property* metaclass used for CIM qualifiers in Qualifier Type  
 1133 Stereotypes, as defined in Table 12.

1134 Table 52 defines the mapping for the attributes and associations of the UML *LiteralString* metaclass for  
 1135 these usages.

1136

Table 52 – UML *LiteralString* Metaclass

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
value	String [0..1]	LiteralString	Shall be the string value to be represented. If the <i>value</i> attribute has no value, the string shall be interpreted as an empty string.
name	String [0]	NamedElement	Compliance value: no value.
visibility	VisibilityKind	PackagableElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
type	Type [0]	TypedElement	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.

### 1137 5.16.3 UML *LiteralInteger* Metaclass

1138 The UML *LiteralInteger* metaclass as defined in this subclause is used to represent integer values in the  
 1139 following UML metaclasses:

- 1140 • *defaultValue* and *lowerValue* associations in UML *Property* metaclass used for CIM properties,  
 1141 as defined in Table 4.
- 1142 • *lowerValue* association in UML *Property* metaclass used for CIM references, as defined in  
 1143 Table 5.
- 1144 • *lowerValue* association in UML *Parameter* metaclass used for CIM parameters and CIM return  
 1145 values, as defined in Table 7.
- 1146 • *value* association in UML *Slot* metaclass used for property values of modeled CIM instances, as  
 1147 defined in Table 10.
- 1148 • *defaultValue* and *lowerValue* associations in UML *Property* metaclass used for CIM qualifiers in  
 1149 Qualifier Type Stereotypes, as defined in Table 12.

1150 Table 53 defines the mapping for the attributes and associations of the UML *LiteralInteger* metaclass for  
 1151 these usages.

1152

Table 53 – UML *LiteralInteger* Metaclass

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
value	Integer	LiteralInteger	Shall be the integer value to be represented.
name	String [0]	NamedElement	Compliance value: no value.
visibility	VisibilityKind	PackagableElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
type	Type [0]	TypedElement	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.

1153 **5.16.4 UML *LiteralUnlimitedNatural* Metaclass**

1154 The UML *LiteralInteger* metaclass as defined in this subclause is used to unlimited natural values in the  
 1155 following UML metaclasses:

- 1156 • *upperValue* association in UML *Property* metaclass used for CIM properties, as defined in  
 1157 Table 4.
- 1158 • *upperValue* association in UML *Property* metaclass used for CIM references, as defined in  
 1159 Table 5.
- 1160 • *upperValue* association in UML *Parameter* metaclass used for CIM parameters and CIM return  
 1161 values, as defined in Table 7.
- 1162 • *upperValue* association in UML *Property* metaclass used for CIM qualifiers in Qualifier Type  
 1163 Stereotypes, as defined in Table 12.

1164 Table 54 defines the mapping for the attributes and associations of the UML *LiteralUnlimitedNatural*  
 1165 metaclass for these usages.

1166 **Table 54 – UML *LiteralUnlimitedNatural* Metaclass**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
value	UnlimitedNatural	LiteralUnlimitedNatural	Shall be the unlimited natural value to be represented.
name	String [0]	NamedElement	Compliance value: no value.
visibility	VisibilityKind	PackagableElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
type	Type [0]	TypedElement	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.

1167 **5.16.5 UML *LiteralNull* Metaclass**

1168 The UML *LiteralNull* metaclass as defined in this subclause is used to represent CIM NULL values in the  
 1169 following UML metaclasses:

- 1170 • *defaultValue* association in UML *Property* metaclass used for CIM properties, as defined in  
 1171 Table 4,
- 1172 • *value* association in UML *Slot* metaclass used for property values of modeled CIM instances, as  
 1173 defined in Table 10,
- 1174 • *defaultValue* association in UML *Property* metaclass used for CIM qualifiers in Qualifier Type  
 1175 Stereotypes, as defined in Table 12.

1176 Table 55 defines the mapping for the attributes and associations of the UML *LiteralNull* metaclass for  
 1177 these usages.

1178

Table 55 – UML *LiteralNull* Metaclass

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
name	String [0]	NamedElement	Compliance value: no value.
visibility	VisibilityKind	PackagableElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
type	Type [0]	TypedElement	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.

1179 **5.16.6 UML Generalization Metaclass**

1180 The UML *Generalization* metaclass as defined in this subclause is used by the following UML  
1181 metaclasses:

- 1182 • *generalization* association in UML *Class* metaclass used for CIM classes and indications,
- 1183 • *generalization* association in UML *AssociationClass* metaclass used for CIM association  
1184 classes.
- 1185 • *generalization* association in UML *Enumeration* metaclass used for CIM *ValueMap* qualifier  
1186 values.

1187 Table 56 defines the mapping for the attributes and associations of the UML *Generalization* metaclass for  
1188 these usages.

1189

Table 56 – UML *Generalization* Metaclass

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
isSubstitutable	Boolean [1]	Generalization	Compliance value: true.  NOTE: UML 2.1.1 does not define a default value for this attribute, OMG issue #9665 has been raised, targeting UML 2.2.
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
general	Classifier	Generalization	Shall be associated with the general <i>Classifier</i> as defined where the <i>Generalization</i> metaclass is used.
specific	Classifier	Generalization	Shall be associated with the specific <i>Classifier</i> as defined where the <i>Generalization</i> metaclass is used.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.

1190 NOTE: The color and other graphical appearance of the generalization link are not normatively defined in this  
1191 document, because the UML Superstructure Specification does not allow specifying the graphical appearance. It is  
1192 recommended that tools apply the DMTF conventions for the color of generalization links (i.e., blue).

1193 **5.16.7 UML *DataType* Metaclass**

1194 The UML *DataType* metaclass as defined in this subclause is used by the following metaclasses:

- 1195 • *type* association in UML *Property* metaclass used for CIM properties and references,
- 1196 • *type* association in UML *Parameter* metaclass used for CIM parameters.

1197 Table 57 defines the mapping for the attributes and associations of the UML *DataType* metaclass for  
 1198 these usages.

1199 **Table 57 – UML *DataType* Metaclass**

UML Attribute	UML Type	Defined in UML Metaclass	CIM Mapping
isAbstract	Boolean	Classifier	Compliance value: false (UML default).
isLeaf	Boolean	RedefinableElement	Compliance value: false (UML default).
name	String	NamedElement	Shall be the name of the CIM datatype.
visibility	VisibilityKind	PackagableElement	Compliance value: public (UML default is private).
UML Association	UML Type	Defined in UML Metaclass	CIM Mapping
ownedAttribute	Property [0]	DataType	Compliance value: no associated elements.
ownedOperation	Operation [0]	DataType	Compliance value: no associated elements.
generalization	Generalization [0]	Classifier	Compliance value: no associated elements.
package	Package [0..1]	Classifier	Shall be owned by and associated with the UML <i>Package</i> metaclass instance representing the type library named "CIMDatatypes", as defined in 5.21.3.
redefinedClassifier	Classifier [0]	Classifier	Compliance value: no associated elements.
elementImport	ElementImport [0]	Namespace	Compliance value: no associated elements.
ownedRule	Constraint [*]	Namespace	No compliance value defined, extension point.
packageImport	PackageImport [0]	Namespace	Compliance value: no associated elements.
ownedComment	Comment [0]	Element	Compliance value: no associated elements.

1200 **5.17 Constraints**

1201 This subclause defines constraints that shall be met in order for a UML user model to be a valid CIM  
 1202 model. These constraints fall into three categories:

- 1203 • OCL constraints for CIM – Constraints specific to the UML profile for CIM, defined using Object  
 1204 Constraint Language (OCL). This is the main category of constraints.
- 1205 • Other constraints for CIM – Constraints specific to the UML profile for CIM, defined using  
 1206 normative text. This category only exists for constraints for which it was not possible to define  
 1207 an according OCL statement.
- 1208 • Constraints for UML – general UML constraints not specific to the UML profile for CIM. They are  
 1209 only listed in order to show which CIM rules are already covered by UML.

1210 A conforming implementation of the UML profile for CIM shall support all OCL constraints for CIM and  
1211 should support all other constraints for CIM.

1212 NOTE: OCL is used as a specification language in this document. Conforming implementations of the UML profile for  
1213 CIM may use other OCL statements or constraint implementations using environments other than OCL as long as  
1214 they produce an equivalent result.

1215 The OCL language used in the definition of the OCL constraints for CIM is defined in the [OCL](#)  
1216 [Specification](#). In addition, some OCL query functions are used as defined in the [UML Superstructure](#)  
1217 [Specification](#). Any remaining OCL functions used in this subclause are defined in 5.17.1.

1218 All OCL constraints for CIM specify the UML element owning the UML *Constraint* metaclass instance  
1219 containing the OCL statement. In some cases, these are specific stereotypes defined in this document. In  
1220 the remaining cases, these are stereotypes defined by the implementation of the UML profile for CIM and  
1221 the context is then defined in terms of UML metaclasses. For example, the context "Stereotype extending  
1222 *Class*" means that "self" in the OCL statement refers to a stereotype that extends the UML *Class*  
1223 metaclass.

1224 All OCL statements in this subclause are invariants.

1225 NOTE 1: A note on accessing the metaclass instance to which a stereotype is applied, from within OCL constraints  
1226 on such stereotypes: According to OMG, the term "self" in such OCL constraints refers to the applied stereotype  
1227 and not to the metaclass instance to which the stereotype is applied. This means that the "base\_<metaclass>"  
1228 relationship needs to be traversed explicitly in OCL statements in order to access the metaclass instance to which the  
1229 stereotype is applied. Some UML tools however define "self" to refer to a merge of the applied stereotype and the  
1230 metaclass instance to which the stereotype is applied. This allows omitting the "base\_<metaclass>" relationship  
1231 traversal in the OCL statement. The OCL statements used in this document follow the OMG direction and use explicit  
1232 traversal of the "base\_<metaclass>" relationship.

1233 NOTE 2: As defined in the [CIM Infrastructure Specification](#), the term "CIM classes" includes CIM associations and  
1234 CIM indications in addition to "normal" CIM classes, and the term "CIM properties" includes CIM references in  
1235 addition to "normal" properties.

1236 NOTE 3: The OCL language allows traversing associations between UML metaclasses in both directions, regardless  
1237 of whether or not the association ends are owned by the association or the associated class. However, OCL engines  
1238 may only support traversal from association ends owned by the associated UML metaclasses, but not from ends  
1239 owned by the associations themselves. The OCL in any constraints defined in this document accomodates for such  
1240 OCL engines in that it only traverses association ends owned by the associated UML metaclasses.

1241 Any ABNF rules defined in the remainder of this subclause use the conventions defined in 3.4.

## 1242 5.17.1 Additional OCL Functions

1243 For the purpose of simplifying the specification of constraints using OCL, some OCL statements use the  
1244 following OCL functions in addition to the OCL syntax defined in the [OCL Specification](#).

### 1245 5.17.1.1 matchCimAbnf

```
1246 String::matchCimAbnf( cimAbnfRuleName : String ) : Boolean
```

1247 Matches the target string against the ABNF rule whose name is specified in `cimAbnfRuleName` and  
1248 returns `true` if the target string matches the format, else `false`. The ABNF rules referenced in  
1249 `cimAbnfRuleName` shall be in the set of rules defined in this document or in the set of rules defined in  
1250 the [CIM Infrastructure Specification](#).

1251 If `cimAbnfRuleName` does not reference a valid ABNF rule, the function returns `null`.

### 1252 5.17.1.2 extractCimAbnf

```
1253 String::extractCimAbnf( cimAbnfRuleName : String,  
1254     extractedCimAbnfRuleName : String ) : String
```



1255 Extracts the substring from the target string that matches the ABNF rule named in  
 1256 `extractedCimAbnfRuleName` and returns the extracted substring. The target string shall match the  
 1257 ABNF rule named in `cimAbnfRuleName`. The ABNF rules referenced in `extractedCimAbnfRuleName`  
 1258 and `cimAbnfRuleName` shall be in the set of rules defined in this document or in the set of rules defined  
 1259 in the [CIM Infrastructure Specification](#).

1260 If the ABNF rule named in `cimAbnfRuleName` is specified with a multiplicity (i.e., using the star (\*) in the  
 1261 ABNF), a list of the matching substrings is returned by the OCL function, in the order they are specified in  
 1262 the target string.

1263 If `extractedCimAbnfRuleName` or `cimAbnfRuleName` do not reference a valid ABNF rule, or if  
 1264 `extractedCimAbnfRuleName` references a rule that is not used in the ABNF rule referenced in  
 1265 `cimAbnfRuleName`, the function returns `null`.

### 1266 5.17.1.3 hexToInteger

1267 `String::hexToInteger() : Integer`

1268 Converts the target string into an integer number. The target string shall match the ABNF format:

1269 `*(hexDigit hexDigit)`

1270 and is further constrained by the value range that can be represented using the OCL Integer datatype. If it  
 1271 does not match that format, or if the value range would be exceeded, the function returns `null`.

### 1272 5.17.1.4 isValidCimReal32

1273 `Integer::isValidCimReal32() : Boolean`

1274 Tests whether the target integer value is a valid CIM real32 number and returns true if that is the case,  
 1275 else false.

### 1276 5.17.1.5 isValidCimReal64

1277 `Integer::isValidCimReal64() : Boolean`

1278 Tests whether the target integer value is a valid CIM real64 number and returns true if that is the case,  
 1279 else false.

### 1280 5.17.1.6 isValidCimString

1281 `String::isValidCimString() : Boolean`

1282 Tests for each character in the string whether it is a Unicode character from the UCS-2 character set and  
 1283 returns true if that is the case, else false.

### 1284 5.17.1.7 isValidCimDatetime

1285 `String::isValidCimDatetime() : Boolean`

1286 Tests whether the string is a valid CIM datetime string and returns true if that is the case, else false.

### 1287 5.17.1.8 isValidCimElementName

1288 `String::isValidCimElementName() : Boolean`

1289 Tests whether the string is a valid CIM element name and returns true if that is the case, else false.

1290 The [CIM Infrastructure Specification](#) defines valid CIM element names as follows:

- 1291 • The name of a CIM element shall consist of the following characters: "\_", "A", ..., "Z", "a", ...,  
1292 "z", "0", ..., "9", U+0080, ..., U+FFEF
- 1293 • The name of a CIM element shall start with one of the following characters: "\_", "A", ..., "Z", "a",  
1294 ..., "z", U+0080, ..., U+FFEF
- 1295 • The name of a CIM element shall be at least one character long.

1296 These rules apply to the names of all CIM element types.

1297 Some CIM element types have additional rules. For example, the name of a CIM class shall have an  
1298 underscore after its schema name part. Such additional rules are validated by additional OCL constraints.

#### 1299 **5.17.1.9 isSet**

1300 `Element::isSet(s : Stereotype, propertyName : String) : Boolean`

1301 Tests whether the stereotype property has a value in the stereotype that is applied to the target element  
1302 and returns true if that is the case, else false.

1303 NOTE: This function is not testing whether the property has a non-default value, nor whether the property is non-null.  
1304 An example where this distinction can be seen is an integer typed stereotype property with a default value of 0. If the  
1305 property has no value set, isSet() returns false. If the property has a value of 0 set isSet() returns true. In both cases,  
1306 a function that returns the value of the property returns 0.

#### 1307 **5.17.1.10 getAppliedStereotype**

1308 `Element::getAppliedStereotype(qualifiedName : String) : Stereotype`

1309 Returns the stereotype with the specified qualified name that is applied to the target element, or null if no  
1310 such stereotype is applied.

1311 The qualified name of a stereotype is its profile name, followed by ":", followed by the stereotype name.

#### 1312 **5.17.1.11 getAllAttributes**

1313 `Classifier::getAllAttributes() : Set(Property)`

1314 Returns the subset of allFeatures() that are properties. Note that this includes inherited properties.

#### 1315 **5.17.1.12 getAllOperations**

1316 `Classifier::getAllOperations() : Set(Operation)`

1317 Returns the subset of allFeatures() that are operations. Note that this includes inherited operations.

#### 1318 **5.17.1.13 getAllSuperClasses**

1319 `Class::getAllSuperClasses() : OrderedSet(Class)`

1320 Returns the set of all superclasses of the target class. The list is ordered from the direct superclasses to  
1321 the most generalized superclasses.

1322 Note that while the definition of this function covers the case of multiple superclasses, a CIM class may  
1323 have at most one superclass (single inheritance).

1324 **5.17.2 OCL Constraints on Classes, Associations, Indications and their Instances**

1325 C1. A non-association CIM class shall have either the *CIM\_Class* or the *CIM\_Indication* stereotype  
 1326 applied.

1327 Context: A stereotype extending UML *Class*

```
1328 OCL: self.base_Class.oclIsTypeOf(uml::AssociationClass) = false
1329     implies
1330     Bag {
1331         self.base_Class.getAppliedStereotype('CIM::CIM_Class'),
1332         self.base_Class.getAppliedStereotype('CIM::CIM_Indication')
1333     }->select( s | s <> null )->size() = 1
```

1334 C2. A CIM association shall have the *CIM\_Association* stereotype applied.

1335 Context: A stereotype extending UML *Association*

```
1336 OCL: self.base_Association.getAppliedStereotype('CIM::CIM_Association')
1337     <> null
```

1338 C3. A CIM association shall be a UML association class (not a plain association link).

1339 Context: A stereotype extending UML *Association*

```
1340 OCL: self.base_Association.oclIsTypeOf(uml::AssociationClass)
```

1341 C4. A CIM class shall have no more than one superclass.

1342 Context: A stereotype extending UML *Class*

```
1343 OCL: self.base_Class.generalization->size() <= 1
```

1344 C5. The superclass of a normal (i.e., non-association, non-indication) CIM class shall be a normal CIM  
 1345 class.

1346 Context: Stereotype *CIM\_Class*

```
1347 OCL: self.base_Class.generalization.general->forall( sc |
1348     sc.getAppliedStereotype('CIM::CIM_Class') <> null )
```

1349 C6. The superclass of a CIM indication shall be a CIM indication.

1350 Context: Stereotype *CIM\_Indication*

```
1351 OCL: self.base_Class.generalization.general->forall( sc |
1352     sc.getAppliedStereotype('CIM::CIM_Indication') <> null )
```

1353 C7. The superclass of a CIM association shall be a CIM association.

1354 Context: Stereotype *CIM\_Association*

```
1355 OCL: self.base_AssociationClass.generalization.general->forall( sc |
1356     sc.getAppliedStereotype('CIM::CIM_Association') <> null )
```

1357 C8. A non-abstract non-indication CIM class shall have key properties.

1358 Context: Stereotype *CIM\_Class*

```
1359 OCL: self.base_Class.isAbstract = false
1360     implies
1361     self.base_Class
1362     ->union(self.base_Class.getAllSuperClasses())
```

```

1363     ->select( c | c.ownedMember
1364         ->any( p | p.getAppliedStereotype('CIM::Key') <> null)->size()>0)
1365     ->size() > 0

```

1366 Context: Stereotype *CIM\_Association*

```

1367 OCL: self.base_AssociationClass.isAbstract = false
1368     implies
1369     self.base_AssociationClass
1370     ->union(self.base_AssociationClass.getAllSuperClasses())
1371     ->select( c | c.ownedMember
1372         ->any( p | p.getAppliedStereotype('CIM::Key') <> null)->size()>0)
1373     ->size() > 0

```

1374 C9. A CIM class with key properties shall not have a superclass with key properties.

1375 NOTE: `getAllAttributes()` returns also any inherited properties

1376 Context: A stereotype extending UML *Class*

```

1377 OCL: let nk : Integer = self.base_Class.ownedAttribute
1378     ->select( p | p.getAppliedStereotype('CIM::Key') <> null)->size()
1379     in
1380     nk > 0
1381     implies
1382     self.base_Class.getAllAttributes()->select( p |
1383         p.getAppliedStereotype('CIM::Key') <> null
1384     )->size() = nk

```

1385 C10. The name of a CIM class shall conform to the format for CIM class names defined in [DSP0004](#).

1386 Represented in ABNF according to 3.4, the format defined in the [CIM Infrastructure Specification](#)  
1387 is:

```

1388     className = schemaName "_" IDENTIFIER;

```

1389 In addition, the `schemaName` shall not include "\_", and both `schemaName` and `IDENTIFIER` shall  
1390 conform to the rules for CIM element names as defined in 5.17.1.8.

1391 Context: A stereotype extending UML *Class*

```

1392 OCL: let cnsz : Integer = /* size of class name */
1393     self.base_Class.name.size()
1394     in
1395     let upos : Integer = /* position of first '_' in class name */
1396         Sequence { 1 .. cnsz }->any( i |
1397             self.base_Class.name.substring( i, i) = '_' )
1398     in
1399     if upos.oclIsUndefined() /* no '_' found */
1400     then false
1401     else
1402         let schema : String = /* schema name of class */
1403             self.base_Class.name.substring( 1, upos - 1)
1404         in
1405         let sname : String = /* short class name */
1406             self.base_Class.name.substring( upos + 1, cnsz)

```

```

1407         in
1408         schema.isValidCimElementName() and
1409         sname.isValidCimElementName()
1410     endif

```

1411 C11. A CIM class shall be in a package path that has *CIM\_Schema* applied somewhere.

1412 Context: A stereotype extending UML *Class*

```

1413 OCL: self.base_Class.allNamespaces()->
1414     select( c | c.getAppliedStereotype('CIM::CIM_Schema') <> null)->
1415     size() > 0

```

1416 C12. The schema name of a CIM class shall be the name of the next outer package that has  
1417 *CIM\_Schema* applied.

1418 Note: As with all CIM names, this comparison is done in a case-insensitive way.

1419 Context: A stereotype extending UML *Class*

```

1420 OCL: let upos : Integer = /* position of first '_' in class name */
1421     Sequence { 1 .. self.base_Class.name.size() }->any( i |
1422         self.base_Class.name.substring( i, i) = '_' )
1423     in
1424     if upos.oclIsUndefined() /* no '_' found */
1425     then false
1426     else
1427         let schema : String = /* schema name of class */
1428             self.base_Class.name.substring( 1, upos - 1)
1429         in
1430         schema.toUpper() = self.base_Class.allNamespaces()->select( c |
1431             c.getAppliedStereotype('CIM::CIM_Schema') <> null)-
1432             >at(1).name.toUpper()
1433     endif

```

1434 C13. All classes in the package subtree under a CIM schema shall have unique names.

1435 Context: Stereotype *CIM\_Schema*, extending any *Package* instance that is supposed to contain an  
1436 entire CIM schema

1437 NOTE 1: A "CIM schema" in this context is any schema, not just the CIM Schema defined by DMTF.

1438 NOTE 2: As with all CIM names, the uniqueness is tested for in a case-insensitive way.

```

1439 OCL: Class.allInstances()->
1440     select(c | c.allNamespaces()->includes(self.base_Package))->
1441     isUnique(c | c.name.toUpper())

```

1442 C14. A CIM class shall not have nested classifiers other than *Enumerations*.

1443 Context: A stereotype extending UML *Class*

```

1444 OCL: self.base_Class.nestedClassifier
1445     ->select( c | c.oclIsTypeOf(uml::Enumeration) = false)
1446     ->isEmpty()

```

- 1447 C15. A CIM class shall have public visibility.  
 1448 Context: A stereotype extending UML *Class*  
 1449 OCL: `self.base_Class.visibility = uml::VisibilityKind::public`
- 1450 C16. A CIM class shall not redefine another classifier.  
 1451 Context: A stereotype extending UML *Class*  
 1452 OCL: `self. base_Class.redefinedClassifier->isEmpty()`
- 1453 C17. A CIM association shall have the UML qualifier `isDerived` set to false.  
 1454 Context: Stereotype *CIM\_Association*  
 1455 OCL: `self. base_AssociationClass.isDerived = false`
- 1456 C18. A generalization shall not have a comment.  
 1457 Context: A stereotype extending UML *Generalization*  
 1458 OCL: `self. base_Generalization.ownedComment->isEmpty()`
- 1459 C19. A generalization shall be substitutable.  
 1460 Context: A stereotype extending UML *Generalization*  
 1461 OCL: `self. base_Generalization.isSubstitutable`
- 1462 C20. A CIM instance shall have public visibility.  
 1463 Context: Stereotype *CIM\_Instance*  
 1464 OCL: `self. base_InstanceSpecification.visibility =`  
 1465 `uml::VisibilityKind::public`
- 1466 C21. A CIM instance shall not have UML specifications.  
 1467 Context: Stereotype *CIM\_Instance*  
 1468 OCL: `self. base_InstanceSpecification.specification->isEmpty()`
- 1469 C22. A CIM class shall not import any elements.  
 1470 Context: A stereotype extending UML *Class*  
 1471 OCL: `self.base_Class.elementImport->isEmpty()`
- 1472 C23. A CIM class shall not import any packages.  
 1473 Context: A stereotype extending UML *Class*  
 1474 OCL: `self.base_Class.packageImport->isEmpty()`

### 1475 5.17.3 OCL Constraints on Properties and References

- 1476 P1. A CIM non-reference property (i.e., not an association end) shall have the *CIM\_Property* stereotype  
 1477 applied.  
 1478 Context: A stereotype extending UML *Property*  
 1479 OCL: `self.base_Property.association->isEmpty()`  
 1480 `implies`  
 1481 `self.base_Property.getAppliedStereotype('CIM::CIM_Property')`  
 1482 `<> null`

- 1483 P2. A CIM reference (i.e., an association end) shall have the *CIM\_Reference* stereotype applied.  
 1484 Context: A stereotype extending UML *Property*  
 1485 OCL: `self.base_Property.association->isEmpty() = false`  
 1486 `implies`  
 1487 `self.base_Property.getAppliedStereotype('CIM::CIM_Reference')`  
 1488 `<> null`
- 1489 P3. A CIM property shall have only one of the stereotypes *CIM\_Property* and *CIM\_Reference* applied.  
 1490 Context: A stereotype extending UML *Property*  
 1491 OCL: `Bag {`  
 1492 `self.base_Property.getAppliedStereotype('CIM::CIM_Property'),`  
 1493 `self.base_Property.getAppliedStereotype('CIM::CIM_Reference')`  
 1494 `}->select( s | s <> null )->size() <> 2`  
 1495 Note: Other constraints validate that normal properties have *CIM\_Property* applied and that  
 1496 association ends have *CIM\_Reference* applied.
- 1497 P4. The end of a CIM association shall be non-navigable.  
 1498 Context: Stereotype *CIM\_Reference*  
 1499 OCL: `self.base_Property.class.getAppliedStereotype(`  
 1500 `'CIM::CIM_Association') <> null`
- 1501 P5. The type of a CIM non-reference property shall be a CIM datatype, an Enumeration or an  
 1502 embedded instance.  
 1503 Context: Meta Element Stereotype *CIM\_Property*  
 1504 OCL: `self.base_Property.type.allNamespaces()->at(1).name =`  
 1505 `'CIMDatatypes' or`  
 1506 `self.base_Property.type.ocIsTypeOf(uml::Enumeration) or`  
 1507 `self.base_Property.type.ocIsTypeOf(uml::Class)`
- 1508 P6. A CIM non-reference property can override only a CIM non-reference property.  
 1509 Context: Stereotype *CIM\_Property*  
 1510 OCL: `self.base_Property.redefinedProperty->isEmpty() = false`  
 1511 `implies (`  
 1512 `self.base_Property.redefinedProperty->size() = 1`  
 1513 `and (`  
 1514 `let rp : Property = self.base_Property.redefinedProperty->`  
 1515 `asSequence()->at(1)`  
 1516 `in`  
 1517 `rp.getAppliedStereotype('CIM::CIM_Property') <> null`  
 1518 `)`  
 1519 `)`

1520 P7. The name of an overridden CIM non-reference property shall not change in the subclass.

1521 NOTE: As with all CIM names, this is tested for in a case-insensitive way.

1522 Context: Stereotype *CIM\_Property*

```

1523 OCL: self.base_Property.redefinedProperty->isEmpty() = false
1524     implies (
1525         self.base_Property.redefinedProperty->size() = 1
1526         and (
1527             let rp : Property = self.base_Property.redefinedProperty->
1528                 asSequence()->at(1)
1529             in
1530                 self.base_Property.name.toUpper() = rp.name.toUpper()
1531         )
1532     )

```

1533 P8. A CIM reference can override only a CIM reference.

1534 Context: Stereotype *CIM\_Reference*

```

1535 OCL: self.base_Property.redefinedProperty->isEmpty() = false
1536     implies (
1537         self.base_Property.redefinedProperty->size() = 1
1538         and (
1539             let rp : Property = self.base_Property.redefinedProperty->
1540                 asSequence()->at(1)
1541             in
1542                 rp.getAppliedStereotype('CIM::CIM_Reference') <> null
1543         )
1544     )

```

1545 P9. The name of an overridden CIM reference shall not change in the subclass.

1546 NOTE: As with all CIM names, this is tested for in a case-insensitive way.

1547 Context: Stereotype *CIM\_Reference*

```

1548 OCL: self.base_Property.redefinedProperty->isEmpty() = false
1549     implies (
1550         self.base_Property.redefinedProperty->size() = 1
1551         and (
1552             let rp : Property = self.base_Property.redefinedProperty->
1553                 asSequence()->at(1)
1554             in
1555                 self.base_Property.name.toUpper() = rp.name.toUpper()
1556         )
1557     )

```



1558 P10. The type of a CIM reference in a subclass shall be the class referenced in the superclass or one of  
1559 its subclasses.

1560 NOTE 1: As with all CIM names, equality test for names is done in a case-insensitive way.

1561 NOTE 2: The association ends in the derived UML AssociationClass exist regardless of whether or not an  
1562 overridden CIM reference in the derived CIM association was defined.

1563 Context: Stereotype *CIM\_Reference*

```
1564 OCL: self.base_Property.class.superClass->isEmpty() = false
1565     implies (
1566         let superp : Property =
1567             self.base_Property.class.superClass.getAllAttributes()->
1568             select( p | p.name.toUpper()
1569                 = self.base_Property.name.toUpper()->
1570                 asSequence()->at(1)
1571             in
1572             self.base_Property.type.conformsTo(superp.type)
1573     )
```

1574 P11. A CIM reference shall not reference a CIM indication.

1575 Context: Stereotype *CIM\_Reference*

```
1576 OCL: self.base_Property.type.getAppliedStereotype('CIM::CIM_Indication') = null
```

1577 P12. A CIM property shall have public visibility.

1578 Context: A stereotype extending UML *Property*

```
1579 OCL: self.base_Property.visibility = uml::VisibilityKind::public
```

1580 P13. A CIM property shall have the UML qualifier *isDerived* set to false.

1581 Context: A stereotype extending UML *Property*

```
1582 OCL: self.base_Property.isDerived = false
```

1583 P14. A CIM property shall have the UML qualifier *isDerivedUnion* set to false.

1584 Context: A stereotype extending UML *Property*

```
1585 OCL: self.base_Property.isDerivedUnion = false
```

1586 P15. A CIM reference shall not be a leaf.

1587 Context: Stereotype *CIM\_Reference*

```
1588 OCL: self.base_Property.isLeaf = false
```

1589 P16. A CIM non-reference property shall not be unique.

1590 Context: Stereotype *CIM\_Property*

```
1591 OCL: self.base_Property.isUnique = false
```

1592 P17. A CIM reference shall be unique unless there are non-reference key properties defined in the same  
1593 class.

1594 Context: Stereotype *CIM\_Reference*

```
1595 OCL: if self.base_Property.class.getAllAttributes()->select( p |
1596     p.getAppliedStereotype('CIM::Key') <> null
1597     and
1598     p.getAppliedStereotype('CIM::CIM_Property') <> null
1599     )->size() = 0
1600 then
1601     self.base_Property.isUnique = false
1602 else
1603     self.base_Property.isUnique = true
1604 endif
```

1605 P18. A CIM property shall not define subsetted properties.

1606 Context: A stereotype extending UML *Property*

```
1607 OCL: self.base_Property.subsettedProperty->isEmpty()
```

1608 P19. A CIM property shall not define UML qualifiers.

1609 Context: A stereotype extending UML *Property*

```
1610 OCL: self.base_Property.qualifier->isEmpty()
```

1611 P20. A CIM reference shall not be static.

1612 Context: Stereotype *CIM\_Reference*

```
1613 OCL: self.base_Property.isStatic = false
```

1614 P21. A CIM reference shall not be ordered.

1615 Context: Stereotype *CIM\_Reference*

```
1616 OCL: self.base_Property.isOrdered = false
```

1617 P22. A CIM key property shall not be an array.

1618 Context: Stereotype *CIM\_Property*

```
1619 OCL: self.base_Property.getAppliedStereotype('CIM::Key') <> null
1620 implies
1621     self.base_Property.upper = 1
```

1622 P23. The name of a CIM property shall conform to the format for CIM element names defined in  
1623 [DSP0004](#).

1624 The format for CIM element names is described in 5.17.1.8.

1625 Context: A stereotype extending UML *Property*

```
1626 OCL: self.base_Property.name.isValidCimElementName()
```

1627 **5.17.4 OCL Constraints on Methods and Parameters**

1628 M1. The name of an overridden CIM method shall not change in the subclass.

1629 NOTE: As with all CIM names, this is tested for in a case-insensitive way.

1630 Context: Stereotype *CIM\_Method*

```

1631 OCL: self.base_Operation.redefinedOperation->isEmpty() = false
1632     implies (
1633         self.base_Operation.redefinedOperation->size() = 1
1634         and (
1635             let ro : Operation = self.base_Operation.redefinedOperation->
1636                 asSequence()->at(1) /* the overridden method */
1637             in
1638                 self.base_Operation.name.toUpper() = ro.name.toUpper()
1639                 /* method name */
1640         )
1641     )

```

1642 M2. The signature of an overridden CIM method shall not change in the subclass.

1643 NOTE 1: The return type is part of the signature.

1644 NOTE 2: As with all CIM names, the equality test for method names is done in a case-insensitive way.

1645 Context: Stereotype *CIM\_Method*

```

1646 OCL: self.base_Operation.redefinedOperation->isEmpty() = false
1647     implies (
1648         self.base_Operation.redefinedOperation->size() = 1
1649         and (
1650             let ro : Operation = self.base_Operation.redefinedOperation->
1651                 asSequence()->at(1) /* the overridden method */
1652             in
1653                 self.base_Operation.type = ro.type /* method return type */
1654             and
1655                 self.base_Operation.ownedParameter->size() =
1656                 ro.ownedParameter->size() /* number of parameters */
1657             and
1658                 Set {1 .. ro.ownedParameter->size()}->forall( i |
1659                     self.base_Operation.ownedParameter->asOrderedSet()->
1660                         at(i).name.toUpper() = ro.ownedParameter->asOrderedSet()->
1661                         at(i).name.toUpper() /* parameter name */
1662                     and
1663                     self.base_Operation.ownedParameter->asOrderedSet()->
1664                         at(i).type = ro.ownedParameter->asOrderedSet()->
1665                         at(i).type /* parameter type */
1666                 )
1667             )
1668     )

```

- 1669 M3. A CIM method shall not be a query.  
 1670 Context: Stereotype *CIM\_Method*  
 1671 OCL: `self.base_Operation.isQuery = false`
- 1672 M4. A CIM method shall have public visibility.  
 1673 Context: Stereotype *CIM\_Method*  
 1674 OCL: `self.base_Operation.visibility = uml::VisibilityKind::public`
- 1675 M5. A CIM method shall not be a leaf.  
 1676 Context: Stereotype *CIM\_Method*  
 1677 OCL: `self.base_Operation.isLeaf = false`
- 1678 M6. A CIM method shall not use body constraints.  
 1679 Context: Stereotype *CIM\_Method*  
 1680 OCL: `self.base_Operation.bodyCondition->isEmpty()`
- 1681 M7. A CIM method shall not define exceptions.  
 1682 Context: Stereotype *CIM\_Method*  
 1683 OCL: `self.base_Operation.raisedException->isEmpty()`
- 1684 M8. A CIM method shall not import any elements.  
 1685 Context: Stereotype *CIM\_Method*  
 1686 OCL: `self.base_Operation.elementImport->isEmpty()`
- 1687 M9. A CIM method shall not import any packages.  
 1688 Context: Stereotype *CIM\_Method*  
 1689 OCL: `self.base_Operation.packageImport->isEmpty()`
- 1690 M10. An in/out/inout parameter of a CIM method shall have the *CIM\_Parameter* stereotype applied.  
 1691 Context: A stereotype extending UML *Parameter*  
 1692 OCL: `self.base_Parameter.direction <>uml::ParameterDirectionKind::return`  
 1693 `implies`  
 1694 `self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter') <> null`
- 1695 M11. A CIM method return value shall not have the *CIM\_Parameter* stereotype applied.  
 1696 Context: A stereotype extending UML *Parameter*  
 1697 OCL: `self.base_Parameter.direction = uml::ParameterDirectionKind::return`  
 1698 `implies`  
 1699 `self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter') = null`
- 1700 M12. The type of a CIM parameter shall be a CIM datatype, an Enumeration or an embedded instance.  
 1701 Context: Stereotype *CIM\_Parameter*  
 1702 OCL: `self.base_Parameter.type.allNamespaces()->at(1).name = 'CIMDatatypes' or`  
 1703 `self.base_Parameter.type.oclIsTypeOf(uml::Enumeration) or`  
 1704 `self.base_Parameter.type.oclIsTypeOf(uml::Class)`

1705 M13. The type of a CIM method return value shall be a CIM datatype, an Enumeration or an embedded  
1706 instance.

1707 NOTE: Returning CIM references is not allowed in CIM.

1708 Context: A stereotype extending UML *Parameter*

1709 OCL: `self.base_Parameter.direction = uml::ParameterDirectionKind::return`  
1710 `implies`  
1711 `self.base_Parameter.type.allNamespaces()->at(1).name = 'CIMDatatypes' or`  
1712 `self.base_Parameter.type.ocIsTypeOf(uml::Enumeration) or`  
1713 `self.base_Parameter.type.ocIsTypeOf(uml::Class)`

1714 M14. A CIM method return value shall not be an array.

1715 Context: A stereotype extending UML *Parameter*

1716 OCL: `self.base_Parameter.direction = uml::ParameterDirectionKind::return`  
1717 `implies`  
1718 `self.base_Parameter.upper = 1`

1719 M15. A CIM parameter shall not have a default value.

1720 Context: Stereotype *CIM\_Parameter*

1721 OCL: `self.base_Parameter.defaultValue->isEmpty()`

1722 M16. A CIM parameter shall have public visibility.

1723 Context: Stereotype *CIM\_Parameter*

1724 OCL: `self.base_Parameter.visibility = uml::VisibilityKind::public`

1725 M17. A CIM parameter shall not be unique.

1726 Context: Stereotype *CIM\_Parameter*

1727 OCL: `self.base_Parameter.isUnique = false`

1728 M18. The name of a CIM method shall conform to the format for CIM element names defined in  
1729 [DSP0004](#).

1730 The format for CIM element names is described in 5.17.1.8.

1731 Context: Stereotype *CIM\_Method*

1732 OCL: `self.base_Operation.name.isValidCimElementName()`

1733 M19. The name of a CIM parameter shall conform to the format for CIM element names defined in  
1734 [DSP0004](#).

1735 The format for CIM element names is described in 5.17.1.8.

1736 Context: Stereotype *CIM\_Parameter*

1737 OCL: `self.base_Parameter.name.isValidCimElementName()`

### 1738 5.17.5 OCL Constraints on Qualifiers

1739 Q1. A CIM qualifier shall have a CIM datatype.

1740 Context: Stereotype *CIM\_QualifierType*

1741 OCL: `self.base_Property.type.allNamespaces()->at(1).name = 'CIMDatatypes'`

- 1742 Q2. A CIM qualifier type stereotype shall have public visibility.  
1743 Context: Stereotype *CIM\_QualifierType*  
1744 OCL: `self.base_Property.class.visibility = uml::VisibilityKind::public`
- 1745 Q3. A CIM qualifier type stereotype shall be abstract.  
1746 Context: Stereotype *CIM\_QualifierType*  
1747 OCL: `self.base_Property.class.isAbstract`
- 1748 Q4. A CIM qualifier type stereotype shall not be a leaf.  
1749 Context: Stereotype *CIM\_QualifierType*  
1750 OCL: `self.base_Property.class.isLeaf = false`
- 1751 Q5. A CIM qualifier type stereotype shall not own UML operations.  
1752 Context: Stereotype *CIM\_QualifierType*  
1753 OCL: `self.base_Property.class.ownedOperation->isEmpty()`
- 1754 Q6. A CIM qualifier type stereotype shall not have a superclass.  
1755 Context: Stereotype *CIM\_QualifierType*  
1756 OCL: `self.base_Property.class.superClass->isEmpty()`
- 1757 Q7. A CIM qualifier type stereotype shall not import any elements.  
1758 Context: Stereotype *CIM\_QualifierType*  
1759 OCL: `self.base_Property.class.elementImport->isEmpty()`
- 1760 Q8. A CIM qualifier type stereotype shall not import any packages.  
1761 Context: Stereotype *CIM\_QualifierType*  
1762 OCL: `self.base_Property.class.packageImport->isEmpty()`
- 1763 Q9. A CIM qualifier type stereotype shall not have a comment.  
1764 Context: Stereotype *CIM\_QualifierType*  
1765 OCL: `self.base_Property.class.ownedComment->isEmpty()`
- 1766 Q10. A CIM qualifier type stereotype shall have one property defined.  
1767 Context: Stereotype *CIM\_QualifierType*  
1768 OCL: `self.base_Property.class.ownedAttribute->size() = 1`
- 1769 Q11. A CIM qualifier type stereotype property shall not be an association end.  
1770 Context: Stereotype *CIM\_QualifierType*  
1771 OCL: `self.base_Property.owningAssociation->isEmpty()`
- 1772 Q12. A CIM qualifier type stereotype property shall have public visibility.  
1773 Context: Stereotype *CIM\_QualifierType*  
1774 OCL: `self.base_Property.visibility = uml::VisibilityKind::public`

- 1775 Q13. A CIM qualifier type stereotype property shall have the UML qualifier *isDerived* set to false.  
 1776 Context: Stereotype *CIM\_QualifierType*  
 1777 OCL: `self.base_Property.isDerived = false`
- 1778 Q14. A CIM qualifier type stereotype property shall have the UML qualifier *isDerivedUnion* set to false.  
 1779 Context: Stereotype *CIM\_QualifierType*  
 1780 OCL: `self.base_Property.isDerivedUnion = false`
- 1781 Q15. A CIM qualifier type stereotype property shall not be read-only.  
 1782 Context: Stereotype *CIM\_QualifierType*  
 1783 OCL: `self.base_Property.isReadOnly = false`
- 1784 Q16. A CIM qualifier type stereotype property shall not be static.  
 1785 Context: Stereotype *CIM\_QualifierType*  
 1786 OCL: `self.base_Property.isStatic = false`
- 1787 Q17. A CIM qualifier type stereotype property shall not be a leaf.  
 1788 Context: Stereotype *CIM\_QualifierType*  
 1789 OCL: `self.base_Property.isLeaf = false`
- 1790 Q18. A CIM qualifier type stereotype property shall be unique.  
 1791 Context: Stereotype *CIM\_QualifierType*  
 1792 OCL: `self.base_Property.isUnique`
- 1793 Q19. A CIM qualifier type stereotype property of array type shall be ordered.  
 1794 Context: Stereotype *CIM\_QualifierType*  
 1795 OCL: `self.base_Property.upper > 1`  
 1796 `implies`  
 1797 `self.base_Property.isOrdered = true`
- 1798 Q20. A CIM qualifier type stereotype property shall not be owned by a structured datatype.  
 1799 Context: Stereotype *CIM\_QualifierType*  
 1800 OCL: `self.base_Property.datatype->isEmpty()`
- 1801 Q21. A CIM qualifier type stereotype property shall not define subsetted properties.  
 1802 Context: Stereotype *CIM\_QualifierType*  
 1803 OCL: `self.base_Property.subsettedProperty->isEmpty()`
- 1804 Q22. A CIM qualifier type stereotype property shall not redefine another property.  
 1805 Context: Stereotype *CIM\_QualifierType*  
 1806 OCL: `self.base_Property.redefinedElement->isEmpty()`
- 1807 Q23. A CIM qualifier type stereotype property shall not define UML qualifiers.  
 1808 Context: Stereotype *CIM\_QualifierType*  
 1809 OCL: `self.base_Property.qualifier->isEmpty()`

1810 Q24. The number of entries in the *BitValues* and *BitMap* qualifier arrays shall match.

1811 Context: Stereotype *CIM\_Property*

1812 OCL: `self.BitMap->size() = self.BitValues->size()`

1813 Context: Stereotype *CIM\_Method*

1814 OCL: `self.BitMap->size() = self.BitValues->size()`

1815 Context: Stereotype *CIM\_Parameter*

1816 OCL: `self.BitMap->size() = self.BitValues->size()`

1817 Q25. The *BitMap* qualifier is applicable only to a CIM integer datatype.

1818 Context: Stereotype *CIM\_Property*

1819 OCL: `let s : Stereotype =`

1820 `self.base_Property.getAppliedStereotype('CIM::CIM_Property')`

1821 `in`

1822 `self.base_Property.isSet(s, 'BitMap')`

1823 `implies`

1824 `Set { /* the integer datatypes */`

1825 `'CIMDatatypes::uint8',`

1826 `'CIMDatatypes::uint16',`

1827 `'CIMDatatypes::uint32',`

1828 `'CIMDatatypes::uint64',`

1829 `'CIMDatatypes::sint8',`

1830 `'CIMDatatypes::sint16',`

1831 `'CIMDatatypes::sint32',`

1832 `'CIMDatatypes::sint64'`

1833 `}->includes(self.base_Property.type.qualifiedName)`

1834 Context: Stereotype *CIM\_Method*

1835 OCL: `let s : Stereotype =`

1836 `self.base_Operation.getAppliedStereotype('CIM::CIM_Method')`

1837 `in`

1838 `self.base_Operation.isSet(s, 'BitMap')`

1839 `implies`

1840 `Set { /* the integer datatypes */`

1841 `'CIMDatatypes::uint8',`

1842 `'CIMDatatypes::uint16',`

1843 `'CIMDatatypes::uint32',`

1844 `'CIMDatatypes::uint64',`

1845 `'CIMDatatypes::sint8',`

1846 `'CIMDatatypes::sint16',`

1847 `'CIMDatatypes::sint32',`

1848 `'CIMDatatypes::sint64'`

1849 `}->includes(self.base_Operation.type.qualifiedName)`



```

1850 Context: Stereotype CIM_Parameter
1851 OCL: let s : Stereotype =
1852     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
1853     in
1854     self.base_Parameter.isSet(s, 'BitMap')
1855     implies
1856     Set { /* the integer datatypes */
1857         'CIMDatatypes::uint8',
1858         'CIMDatatypes::uint16',
1859         'CIMDatatypes::uint32',
1860         'CIMDatatypes::uint64',
1861         'CIMDatatypes::sint8',
1862         'CIMDatatypes::sint16',
1863         'CIMDatatypes::sint32',
1864         'CIMDatatypes::sint64'
1865     }->includes(self.base_Parameter.type.qualifiedName)

```

1866 Q26. The *BitValues* qualifier is applicable only to a CIM integer datatype.

```

1867 Context: Stereotype CIM_Property
1868 OCL: let s : Stereotype =
1869     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
1870     in
1871     self.base_Property.isSet(s, 'BitValues')
1872     implies
1873     Set { /* the integer datatypes */
1874         'CIMDatatypes::uint8',
1875         'CIMDatatypes::uint16',
1876         'CIMDatatypes::uint32',
1877         'CIMDatatypes::uint64',
1878         'CIMDatatypes::sint8',
1879         'CIMDatatypes::sint16',
1880         'CIMDatatypes::sint32',
1881         'CIMDatatypes::sint64'
1882     }->includes(self.base_Property.type.qualifiedName)

```

```

1883 Context: Stereotype CIM_Method
1884 OCL: let s : Stereotype =
1885     self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
1886     in
1887     self.base_Operation.isSet(s, 'BitValues')
1888     implies
1889     Set { /* the integer datatypes */
1890         'CIMDatatypes::uint8',
1891         'CIMDatatypes::uint16',
1892         'CIMDatatypes::uint32',
1893         'CIMDatatypes::uint64',
1894         'CIMDatatypes::sint8',
1895         'CIMDatatypes::sint16',

```

```

1896         'CIMDatatypes::sint32',
1897         'CIMDatatypes::sint64'
1898     }->includes(self.base_Operation.type.qualifiedName)

1899 Context: Stereotype CIM_Parameter

1900 OCL: let s : Stereotype =
1901     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
1902 in
1903 self.base_Parameter.isSet(s, 'BitValues')
1904 implies
1905 Set { /* the integer datatypes */
1906     'CIMDatatypes::uint8',
1907     'CIMDatatypes::uint16',
1908     'CIMDatatypes::uint32',
1909     'CIMDatatypes::uint64',
1910     'CIMDatatypes::sint8',
1911     'CIMDatatypes::sint16',
1912     'CIMDatatypes::sint32',
1913     'CIMDatatypes::sint64'
1914 }->includes(self.base_Parameter.type.qualifiedName)

```

1915 Q27. The *Counter* qualifier is applicable only to a CIM unsigned integer datatype.

```

1916 Context: Stereotype CIM_Property

1917 OCL: let s : Stereotype =
1918     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
1919 in
1920 self.base_Property.isSet(s, 'Counter')
1921 implies
1922 Set { /* the unsigned integer datatypes */
1923     'CIMDatatypes::uint8',
1924     'CIMDatatypes::uint16',
1925     'CIMDatatypes::uint32',
1926     'CIMDatatypes::uint64'
1927 }->includes(self.base_Property.type.qualifiedName)

```

```

1928 Context: Stereotype CIM_Method

1929 OCL: let s : Stereotype =
1930     self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
1931 in
1932 self.base_Operation.isSet(s, 'Counter')
1933 implies
1934 Set { /* the unsigned integer datatypes */
1935     'CIMDatatypes::uint8',
1936     'CIMDatatypes::uint16',
1937     'CIMDatatypes::uint32',
1938     'CIMDatatypes::uint64'
1939 }->includes(self.base_Operation.type.qualifiedName)

```

1940 Context: Stereotype *CIM\_Parameter*

```

1941 OCL: let s : Stereotype =
1942     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
1943     in
1944     self.base_Parameter.isSet(s,'Counter')
1945     implies
1946     Set { /* the unsigned integer datatypes */
1947         'CIMDatatypes::uint8',
1948         'CIMDatatypes::uint16',
1949         'CIMDatatypes::uint32',
1950         'CIMDatatypes::uint64'
1951     }->includes(self.base_Parameter.type.qualifiedName)

```

1952 Q28. A value of the *Deprecated* qualifier shall conform to the format defined in [DSP0004](#).

1953 Represented in ABNF according to 3.4, the format for non-NULL values of the *Deprecated* qualifier  
 1954 as defined in the [CIM Infrastructure Specification](#) and [DSP0004](#) is:

```

1955     DeprecatedFormat = className [ [ embeddedInstancePath ]
1956         "." elementSpec ];
1957     elementSpec = propertyName |
1958         methodName "(" [parameterName *("," parameterName)] ")";
1959     embeddedInstancePath = 1*( "." propertyName );

```

1960 where the non-terminals on the right hand side of these ABNF rules are defined in Appendix A of  
 1961 [DSP0004](#).

1962 Context: Stereotype *CIM\_Class*

1963 OCL: self.Deprecated.matchCimAbnf('DeprecatedFormat')

1964 Context: Stereotype *CIM\_Association*

1965 OCL: self.Deprecated.matchCimAbnf('DeprecatedFormat')

1966 Context: Stereotype *CIM\_Indication*

1967 OCL: self.Deprecated.matchCimAbnf('DeprecatedFormat')

1968 Context: Stereotype *CIM\_Property*

1969 OCL: self.Deprecated.matchCimAbnf('DeprecatedFormat')

1970 Context: Stereotype *CIM\_Reference*

1971 OCL: self.Deprecated.matchCimAbnf('DeprecatedFormat')

1972 Context: Stereotype *CIM\_Method*

1973 OCL: self.Deprecated.matchCimAbnf('DeprecatedFormat')

1974 Context: Stereotype *CIM\_Parameter*

1975 OCL: self.Deprecated.matchCimAbnf('DeprecatedFormat')

- 1976 Q29. If stereotype property *Deprecated* has a value, the marker stereotype *Deprecated* shall be applied.
- 1977 Context: Stereotype *CIM\_Class* and *CIM\_Indication*
- 1978 OCL: `self.Deprecated->size() > 0`
- 1979 `implies`
- 1980 `self.base_Class.getAppliedStereotype('CIM::Deprecated') <> null`
- 1981 Context: Stereotype *CIM\_Association*
- 1982 OCL: `self.Deprecated->size() > 0`
- 1983 `implies`
- 1984 `self.base_AssociationClass.getAppliedStereotype('CIM::Deprecated') <> null`
- 1985 Context: Stereotypes *CIM\_Property* and *CIM\_Reference*
- 1986 OCL: `self.Deprecated->size() > 0`
- 1987 `implies`
- 1988 `self.base_Property.getAppliedStereotype('CIM::Deprecated') <> null`
- 1989 Context: Stereotype *CIM\_Method*
- 1990 OCL: `self.Deprecated->size() > 0`
- 1991 `implies`
- 1992 `self.base_Operation.getAppliedStereotype('CIM::Deprecated') <> null`
- 1993 Context: Stereotype *CIM\_Parameter*
- 1994 OCL: `self.Deprecated->size() > 0`
- 1995 `implies`
- 1996 `self.base_Parameter.getAppliedStereotype('CIM::Deprecated') <> null`
- 1997 Q30. If the *Deprecated* qualifier is specified on the class owning a property, it should also be specified on
- 1998 that property.
- 1999 Context: Stereotypes *CIM\_Property* and *CIM\_Reference*
- 2000 OCL: `self.base_Property.class.getAppliedStereotype('CIM::Deprecated') <> null`
- 2001 `implies`
- 2002 `self.base_Property.getAppliedStereotype('CIM::Deprecated') <> null`
- 2003 Q31. If the *Deprecated* qualifier is specified on the class owning a method, it should also be specified on
- 2004 that method.
- 2005 Context: Stereotype *CIM\_Method*
- 2006 OCL: `self.base_Operation.class.getAppliedStereotype('CIM::Deprecated') <> null`
- 2007 `implies`
- 2008 `self.base_Operation.getAppliedStereotype('CIM::Deprecated') <> null`
- 2009 Q32. The *DN* qualifier is applicable only to a CIM string datatype.
- 2010 Context: Stereotype *CIM\_Property*
- 2011 OCL: `let s : Stereotype =`
- 2012 `self.base_Property.getAppliedStereotype('CIM::CIM_Property')`
- 2013 `in`
- 2014 `self.base_Property.isSet(s, 'DN')`
- 2015 `implies`
- 2016 `Set { /* the string datatypes */`

```

2017         'CIMDatatypes::string'
2018     }->includes(self.base_Property.type.qualifiedName)

```

2019 **Context: Stereotype *CIM\_Method***

```

2020 OCL: let s : Stereotype =
2021     self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
2022     in
2023     self.base_Operation.isSet(s, 'DN')
2024     implies
2025     Set { /* the string datatypes */
2026         'CIMDatatypes::string'
2027     }->includes(self.base_Operation.type.qualifiedName)

```

2028 **Context: Stereotype *CIM\_Parameter***

```

2029 OCL: let s : Stereotype =
2030     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
2031     in
2032     self.base_Parameter.isSet(s, 'DN')
2033     implies
2034     Set { /* the string datatypes */
2035         'CIMDatatypes::string'
2036     }->includes(self.base_Parameter.type.qualifiedName)

```

2037 Q33. The *EmbeddedObject* qualifier is applicable only to a CIM string datatype.

2038 **Context: Stereotype *CIM\_Property***

```

2039 OCL: let s : Stereotype =
2040     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2041     in
2042     self.base_Property.isSet(s, 'EmbeddedObject')
2043     implies
2044     Set { /* the string datatypes */
2045         'CIMDatatypes::string'
2046     }->includes(self.base_Property.type.qualifiedName)

```

2047 **Context: Stereotype *CIM\_Method***

```

2048 OCL: let s : Stereotype =
2049     self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
2050     in
2051     self.base_Operation.isSet(s, 'EmbeddedObject')
2052     implies
2053     Set { /* the string datatypes */
2054         'CIMDatatypes::string'
2055     }->includes(self.base_Operation.type.qualifiedName)

```

2056 **Context: Stereotype *CIM\_Parameter***

```

2057 OCL: let s : Stereotype =
2058     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
2059     in
2060     self.base_Parameter.isSet(s, 'EmbeddedObject')

```

```

2061         implies
2062         Set { /* the string datatypes */
2063             'CIMDatatypes::string'
2064         }->includes(self.base_Parameter.type.qualifiedName)

2065 Q34. A CIM class shall not be abstract and have an Exception qualifier with an effective value of true.
2066 Context: Stereotypes CIM_Class and CIM_Indication
2067 OCL: not (self.base_Class.isAbstract and self.Exception)

2068 Q35. The Gauge qualifier is applicable only to a CIM unsigned integer datatype.
2069 Context: Stereotype CIM_Property
2070 OCL: let s : Stereotype =
2071     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2072 in
2073 self.base_Property.isSet(s, 'Gauge')
2074 implies
2075 Set { /* the unsigned integer datatypes */
2076     'CIMDatatypes::uint8',
2077     'CIMDatatypes::uint16',
2078     'CIMDatatypes::uint32',
2079     'CIMDatatypes::uint64'
2080 }->includes(self.base_Property.type.qualifiedName)

2081 Context: Stereotype CIM_Method
2082 OCL: let s : Stereotype =
2083     self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
2084 in
2085 self.base_Operation.isSet(s, 'Gauge')
2086 implies
2087 Set { /* the unsigned integer datatypes */
2088     'CIMDatatypes::uint8',
2089     'CIMDatatypes::uint16',
2090     'CIMDatatypes::uint32',
2091     'CIMDatatypes::uint64'
2092 }->includes(self.base_Operation.type.qualifiedName)

2093 Context: Stereotype CIM_Parameter
2094 OCL: let s : Stereotype =
2095     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
2096 in
2097 self.base_Parameter.isSet(s, 'Gauge')
2098 implies
2099 Set { /* the unsigned integer datatypes */
2100     'CIMDatatypes::uint8',
2101     'CIMDatatypes::uint16',
2102     'CIMDatatypes::uint32',
2103     'CIMDatatypes::uint64'
2104 }->includes(self.base_Parameter.type.qualifiedName)

```

2105 Q36. A value of an entry in the *MappingStrings* qualifier array shall conform to the format defined in  
2106 [DSP0004](#).

2107 Represented in ABNF according to 3.4, the format for non-NULL values of the *MappingStrings*  
2108 qualifier as defined in [DSP0004](#) is:

```
2109 MappingStringsFormat = mib_format | oid_format | general_format | mif_format;
2110 mib_format = "MIB" "." mib_naming_authority "|" mib_name "." mib_variable_name;
2111 mib_naming_authority = 1*(stringChar);
2112 mib_name = 1*(stringChar);
2113 mib_variable_name = 1*(stringChar);
2114 oid_format = "OID" "." oid_naming_authority "|" oid_protocol_name "." oid;
2115 oid_naming_authority = 1*(stringChar);
2116 oid_protocol_name = 1*(stringChar);
2117 oid = 1*(stringChar);
2118 general_format = general_format_fullname "|" general_format_mapping;
2119 general_format_fullname = general_format_name "." general_format_defining_body;
2120 general_format_name = 1*(stringChar);
2121 general_format_defining_body = 1*(stringChar);
2122 general_format_mapping = 1*(stringChar);
2123 mif_format = mif_attribute_format | mif_group_format;
2124 mif_attribute_format = "MIF" "." mif_class_string "." mif_attribute_id;
2125 mif_group_format = "MIF" "." mif_class_string;
2126 mif_class_string = mif_defining_body "|" mif_specific_name "|" mif_version;
2127 mif_defining_body = 1*(stringChar);
2128 mif_specific_name = 1*(stringChar);
2129 mif_version = 3(decimalDigit);
2130 mif_attribute_id = positiveDecimalDigit *decimalDigit;
```

2131 where the non-terminals on the right hand side of these ABNF rules are defined in Appendix A of  
2132 [DSP0004](#).

2133 Context: Stereotypes *CIM\_Class*, *CIM\_Association*, *CIM\_Indication*, *CIM\_Property*,  
2134 *CIM\_Reference*, *CIM\_Method*, *CIM\_Parameter*

2135 OCL: self.MappingStrings->forall( qv | qv.matchCimAbnf( 'MappingStringsFormat' )  
2136 )

2137 Q37. The *MaxLen* qualifier is applicable only to a CIM string datatype.

2138 Context: Stereotype *CIM\_Property*

```
2139 OCL: let s : Stereotype =
2140     self.base_Property.getAppliedStereotype( 'CIM::CIM_Property' )
2141     in
2142     self.base_Property.isSet( s, 'MaxLen' )
2143     implies
2144     Set { /* the string datatypes */
2145         'CIMDatatypes::string'
2146     }->includes( self.base_Property.type.qualifiedName )
```

```

2147     Context: Stereotype CIM_Method
2148     OCL: let s : Stereotype =
2149         self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
2150     in
2151     self.base_Operation.isSet(s, 'MaxLen')
2152     implies
2153     Set { /* the string datatypes */
2154         'CIMDatatypes::string'
2155     }->includes(self.base_Operation.type.qualifiedName)

2156     Context: Stereotype CIM_Parameter
2157     OCL: let s : Stereotype =
2158         self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
2159     in
2160     self.base_Parameter.isSet(s, 'MaxLen')
2161     implies
2162     Set { /* the string datatypes */
2163         'CIMDatatypes::string'
2164     }->includes(self.base_Parameter.type.qualifiedName)

2165 Q38. The MinLen qualifier is applicable only to a CIM string datatype.
2166     Context: Stereotype CIM_Property
2167     OCL: let s : Stereotype =
2168         self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2169     in
2170     self.base_Property.isSet(s, 'MinLen')
2171     implies
2172     Set { /* the string datatypes */
2173         'CIMDatatypes::string'
2174     }->includes(self.base_Property.type.qualifiedName)

2175     Context: Stereotype CIM_Method
2176     OCL: let s : Stereotype =
2177         self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
2178     in
2179     self.base_Operation.isSet(s, 'MinLen')
2180     implies
2181     Set { /* the string datatypes */
2182         'CIMDatatypes::string'
2183     }->includes(self.base_Operation.type.qualifiedName)

2184     Context: Stereotype CIM_Parameter
2185     OCL: let s : Stereotype =
2186         self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
2187     in
2188     self.base_Parameter.isSet(s, 'MinLen')
2189     implies
2190     Set { /* the string datatypes */

```



```

2191         'CIMDatatypes::string'
2192     }->includes(self.base_Parameter.type.qualifiedName)

```

2193 Q39. The *MaxValue* qualifier is applicable only to a CIM numeric datatype.

2194 Context: Stereotype *CIM\_Property*

```

2195 OCL: let s : Stereotype =
2196     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2197 in
2198 self.base_Property.isSet(s, 'MaxValue')
2199 implies
2200 Set { /* the numeric datatypes */
2201     'CIMDatatypes::uint8',
2202     'CIMDatatypes::uint16',
2203     'CIMDatatypes::uint32',
2204     'CIMDatatypes::uint64',
2205     'CIMDatatypes::sint8',
2206     'CIMDatatypes::sint16',
2207     'CIMDatatypes::sint32',
2208     'CIMDatatypes::sint64',
2209     'CIMDatatypes::real32',
2210     'CIMDatatypes::real64'
2211 }->includes(self.base_Property.type.qualifiedName)

```

2212 Context: Stereotype *CIM\_Method*

```

2213 OCL: let s : Stereotype =
2214     self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
2215 in
2216 self.base_Operation.isSet(s, 'MaxValue')
2217 implies
2218 Set { /* the numeric datatypes */
2219     'CIMDatatypes::uint8',
2220     'CIMDatatypes::uint16',
2221     'CIMDatatypes::uint32',
2222     'CIMDatatypes::uint64',
2223     'CIMDatatypes::sint8',
2224     'CIMDatatypes::sint16',
2225     'CIMDatatypes::sint32',
2226     'CIMDatatypes::sint64',
2227     'CIMDatatypes::real32',
2228     'CIMDatatypes::real64'
2229 }->includes(self.base_Operation.type.qualifiedName)

```

2230 Context: Stereotype *CIM\_Parameter*

```

2231 OCL: let s : Stereotype =
2232     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
2233 in
2234 self.base_Parameter.isSet(s, 'MaxValue')
2235 implies
2236 Set { /* the numeric datatypes */

```

```

2237         'CIMDatatypes::uint8',
2238         'CIMDatatypes::uint16',
2239         'CIMDatatypes::uint32',
2240         'CIMDatatypes::uint64',
2241         'CIMDatatypes::sint8',
2242         'CIMDatatypes::sint16',
2243         'CIMDatatypes::sint32',
2244         'CIMDatatypes::sint64',
2245         'CIMDatatypes::real32',
2246         'CIMDatatypes::real64'
2247     }->includes(self.base_Parameter.type.qualifiedName)

```

2248 Q40. The *MinValue* qualifier is applicable only to a CIM numeric datatype.

2249 Context: Stereotype *CIM\_Property*

```

2250 OCL: let s : Stereotype =
2251     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2252 in
2253 self.base_Property.isSet(s, 'MinValue')
2254 implies
2255 Set { /* the numeric datatypes */
2256     'CIMDatatypes::uint8',
2257     'CIMDatatypes::uint16',
2258     'CIMDatatypes::uint32',
2259     'CIMDatatypes::uint64',
2260     'CIMDatatypes::sint8',
2261     'CIMDatatypes::sint16',
2262     'CIMDatatypes::sint32',
2263     'CIMDatatypes::sint64',
2264     'CIMDatatypes::real32',
2265     'CIMDatatypes::real64'
2266 }->includes(self.base_Property.type.qualifiedName)

```

2267 Context: Stereotype *CIM\_Method*

```

2268 OCL: let s : Stereotype =
2269     self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
2270 in
2271 self.base_Operation.isSet(s, 'MinValue')
2272 implies
2273 Set { /* the numeric datatypes */
2274     'CIMDatatypes::uint8',
2275     'CIMDatatypes::uint16',
2276     'CIMDatatypes::uint32',
2277     'CIMDatatypes::uint64',
2278     'CIMDatatypes::sint8',
2279     'CIMDatatypes::sint16',
2280     'CIMDatatypes::sint32',
2281     'CIMDatatypes::sint64',
2282     'CIMDatatypes::real32',
2283     'CIMDatatypes::real64'
2284 }->includes(self.base_Operation.type.qualifiedName)

```

2285 Context: Stereotype *CIM\_Parameter*

```

2286 OCL: let s : Stereotype =
2287     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
2288     in
2289     self.base_Parameter.isSet(s,'MinValue')
2290     implies
2291     Set { /* the numeric datatypes */
2292         'CIMDatatypes::uint8',
2293         'CIMDatatypes::uint16',
2294         'CIMDatatypes::uint32',
2295         'CIMDatatypes::uint64',
2296         'CIMDatatypes::sint8',
2297         'CIMDatatypes::sint16',
2298         'CIMDatatypes::sint32',
2299         'CIMDatatypes::sint64',
2300         'CIMDatatypes::real32',
2301         'CIMDatatypes::real64'
2302     }->includes(self.base_Parameter.type.qualifiedName)

```

2303 Q41. A value of the *ModelCorrespondence* qualifier shall conform to the format defined in [DSP0004](#).

2304 The format defined in [DSP0004](#) is:

```

2305 className [ *( "." ( propertyName | referenceName ) ) [ "." methodName [ "("
2306     parameterName ")" ] ] ]

```

2307 Note that this format definition is currently flawed for multiple reasons:

- 2308 • It allows only one parameter for a method.
- 2309 • It does not describe in [DSP0004](#) that the concatenation of multiple properties is meant to  
2310 denote elements in embedded properties.
- 2311 • It allows further elements to be specified after a reference as if the reference were an  
2312 embedded property, which does not make sense.
- 2313 • It does not allow distinguishing between methods and properties when they have the same  
2314 name (a rare case, admittedly).

2315 Therefore, this document is using the following improved format definition, which is intended to be  
2316 incorporated into [DSP0004](#) at some point. Represented in ABNF according to 3.4, the format is:

```

2317 ModelCorrespondenceFormat = className [ [ embeddedPropertyNames ]
2318     "." ( propertyName | referenceName |
2319         methodName "(" [ parameterName *( "," parameterName ) ] ")" )
2320     ];

```

```

2321 embeddedPropertyNames = 1*( "." propertyName );

```

2322 where the remaining rules on the right hand side of these ABNF rules are defined in [DSP0004](#),  
2323 Appendix A.

2324 Context: Stereotypes *CIM\_Class*, *CIM\_Association*, *CIM\_Indication*, *CIM\_Property*,  
2325 *CIM\_Reference*, *CIM\_Method* and *CIM\_Parameter*

```

2326 OCL: self.ModelCorrespondence->forall( qv |
2327     qv.matchCimAbnf( 'ModelCorrespondenceFormat' )

```

2328 Q42. The CIM elements referenced in a *ModelCorrespondence* qualifier value shall exist.

2329 NOTE: As with all CIM names, equality tests are done in a case-insensitive way.

2330 Context: Stereotypes *CIM\_Class*, *CIM\_Association*, *CIM\_Indication*, *CIM\_Property*,  
2331 *CIM\_Reference*, *CIM\_Method*, *CIM\_Parameter*

```

2332 OCL: let className : String = self.ModelCorrespondence.extractCimAbnf(
2333     'ModelCorrespondenceFormat', 'className' )
2334     in
2335     let propName : String = self.ModelCorrespondence.extractCimAbnf(
2336         'ModelCorrespondenceFormat', 'propertyName' )
2337         in
2338         let refName : String = self.ModelCorrespondence.extractCimAbnf(
2339             'ModelCorrespondenceFormat', 'referenceName' )
2340             in
2341             let methName : String = self.ModelCorrespondence.extractCimAbnf(
2342                 'ModelCorrespondenceFormat', 'methodName' )
2343                 in
2344                 let parmNames : OrderedSet(String) =
2345                     self.ModelCorrespondence.extractCimAbnf('ModelCorrespondenceFormat',
2346                         'parameterName' )
2347                     in
2348                     let embPropNames : OrderedSet(String) =
2349                         self.ModelCorrespondence.extractCimAbnf('ModelCorrespondenceFormat',
2350                             'embeddedPropertyNames').extractCimAbnf( 'embeddedPropertyNames',
2351                                 'propertyName' )
2352                         in
2353                         let c : Class = Class.allInstances()->select( _c | _c.name.toUpper() =
2354                             className.toUpper())
2355                         in
2356                             c <> null
2357                         and
2358                             propName <> null implies
2359                                 c.getAllAttributes()->select( p | p.name.toUpper() =
2360                                     propName.toUpper()) <> null)
2361                         and
2362                             refName <> null implies
2363                                 c.getAllAttributes()->select( p | p.name.toUpper() =
2364                                     refName.toUpper()) <> null)
2365                         and
2366                             methName <> null implies
2367                                 c.getAllOperations()->select( m | m.name.toUpper() =
2368                                     methName.toUpper())->select( m | m.ownedParameter.name
2369                                     ->asOrderedSet().toUpper() = parmNames.toUpper())
2370                             )
2371                     )

```

2372 Q43. The *NullValue* qualifier is applicable only to CIM string and integer datatypes.

2373 Context: Stereotype *CIM\_Property*

```

2374 OCL: let s : Stereotype =
2375     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2376     in
2377     self.base_Property.isSet(s, 'NullValue')
2378     implies

```

```

2379     Set { /* the string and integer datatypes */
2380         'CIMDatatypes::uint8',
2381         'CIMDatatypes::uint16',
2382         'CIMDatatypes::uint32',
2383         'CIMDatatypes::uint64',
2384         'CIMDatatypes::sint8',
2385         'CIMDatatypes::sint16',
2386         'CIMDatatypes::sint32',
2387         'CIMDatatypes::sint64',
2388         'CIMDatatypes::string',
2389         'CIMDatatypes::char16'
2390     }->includes(self.base_Property.type.qualifiedName)

```

2391 Q44. A value of the *NullValue* qualifier when used with an integer datatype shall conform to the format  
 2392 defined in [DSP0004](#).

2393 Represented in ABNF according to 3.4, the format defined in [DSP0004](#) for this case is:

```

2394 NullValueFormatForIntegers = [ "+" | "-" ] 1*(decimalDigit);

```

2395 where the rule on the right hand side of this ABNF rule is defined in Appendix A of [DSP0004](#), and  
 2396 the value range of this format is constrained by the integer datatype of the CIM element to which  
 2397 this qualifier is applied. For example, the value range for this qualifier when applied to a CIM  
 2398 element with a `uint8` datatype would be 0..127.

2399 Context: Stereotype *CIM\_Property*

```

2400 OCL: let s : Stereotype =
2401     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2402     in
2403     self.base_Property.isSet(s, 'NullValue')
2404     and
2405     Set { /* the integer datatypes */
2406         'CIMDatatypes::uint8',
2407         'CIMDatatypes::uint16',
2408         'CIMDatatypes::uint32',
2409         'CIMDatatypes::uint64',
2410         'CIMDatatypes::sint8',
2411         'CIMDatatypes::sint16',
2412         'CIMDatatypes::sint32',
2413         'CIMDatatypes::sint64'
2414     }->includes(self.base_Property.type.qualifiedName)
2415     implies
2416     self.NullValue.matchCimAbnf( 'NullValueFormatForIntegers' )

```

2417 Q45. A value of the *Propagated* qualifier shall conform to the format defined in [DSP0004](#).

2418 Represented in ABNF according to 3.4, the format defined in [DSP0004](#) is:

```

2419 PropagatedFormat = [className "." ] IDENTIFIER;

```

2420 where the rules on the right hand side of this ABNF rule are defined in Appendix A of [DSP0004](#).

2421 Context: Stereotype *CIM\_Property*

```

2422 OCL: let s : Stereotype =
2423     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2424     in
2425     self.base_Property.isSet(s, 'Propagated')

```

```

2426         implies
2427         self.Propagated.matchCimAbnf( 'PropagatedFormat' )

```

2428 Q46. The property referenced in a *Propagated* qualifier value shall be exposed by the specified class.

2429 NOTE: As with all CIM names, equality tests are done in a case-insensitive way.

2430 Context: Stereotype *CIM\_Property*

```

2431 OCL: let s : Stereotype =
2432         self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2433     in
2434     self.base_Property.isSet(s, 'Propagated')
2435     implies
2436     let className : String = self.Propagated.extractCimAbnf(
2437         'PropagatedFormat', 'className' )
2438     in
2439     let propName : String = self.Propagated.extractCimAbnf(
2440         'PropagatedFormat', 'IDENTIFIER' )
2441     in
2442     let myc : Class = /* class receiving the propagated property */
2443         self.oclAsType(CIM::CIM_Property).base_Property.class
2444     in
2445     let c : Class = /* class that owns the propagated property */
2446         if className = null
2447         then
2448             myc /* default if no class was specified */
2449         else
2450             Class.allInstances()->select( c | c.name.toUpper() =
2451                 className.toUpper()
2452                 ->asOrderedSet()->at(1)
2453         endif
2454     in
2455     let p : Property = /* the propagated property */
2456         c.getAllAttributes()->select( p | p.name.toUpper() = propName.toUpper()
2457     in
2458         c <> null and p <> null

```

2459 Q47. A property qualified with *Propagated* shall be a key property.

2460 Context: Stereotype *CIM\_Property*

```

2461 OCL: let s : Stereotype =
2462         self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2463     in
2464     self.base_Property.isSet(s, 'Propagated')
2465     implies
2466     self.base_Property.getAppliedStereotype('CIM::Key') <> null

```

2467 Q48. The property referenced in a *Propagated* qualifier value shall have the same datatype as the  
2468 qualified property.

2469 NOTE: As with all CIM names, equality tests are done in a case-insensitive way.

2470 Context: Stereotype *CIM\_Property*

```

2471 OCL: let s : Stereotype =
2472         self.base_Property.getAppliedStereotype('CIM::CIM_Property')

```

```

2473     in
2474     self.base_Property.isSet(s,'Propagated')
2475     implies
2476     let className : String = self.Propagated.extractCimAbnf(
2477         'PropagatedFormat', 'className' )
2478     in
2479     let propName : String = self.Propagated.extractCimAbnf(
2480         'PropagatedFormat', 'IDENTIFIER' )
2481     in
2482     let myc : Class = /* class receiving the propagated property */
2483         self.oclAsType(CIM::CIM_Property).base_Property.class
2484     in
2485     let c : Class = /* class that owns the propagated property */
2486         if className = null
2487         then
2488             myc /* default if no class was specified */
2489         else
2490             Class.allInstances()->select( c | c.name.toUpper() =
2491                 className.toUpper()
2492                 ->asOrderedSet()->at(1)
2493             endif
2494     in
2495     let p : Property = /* the propagated property */
2496         c.getAllAttributes()->select( p | p.name.toUpper() = propName.toUpper() )
2497     in
2498     p.type = self.oclAsType(CIM::CIM_Property).base_Property.type

```

Q49. The class of a property qualified with the *Propagated* qualifier and the class of the property referenced in that *Propagated* qualifier value shall have an association defined between them which is qualified as weak on the reference referencing the property qualified with the *Propagated* qualifier.

NOTE: As with all CIM names, equality tests are done in a case-insensitive way.

Context: Stereotype *CIM\_Property*

```

2505 OCL: let s : Stereotype =
2506     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2507 in
2508 self.base_Property.isSet(s,'Propagated')
2509 implies
2510 let className : String = self.Propagated.extractCimAbnf(
2511     'PropagatedFormat', 'className' )
2512 in
2513 let propName : String = self.Propagated.extractCimAbnf(
2514     'PropagatedFormat', 'IDENTIFIER' )
2515 in
2516 let myc : Class = /* class receiving the propagated property */
2517     self.oclAsType(CIM::CIM_Property).base_Property.class
2518 in
2519 let c : Class = /* class that owns the propagated property */
2520     if className = null

```

```

2521         then
2522             myc /* default if no class was specified */
2523         else
2524             Class.allInstances()->select( c | c.name.toUpper() =
2525                 className.toUpper()
2526                 ->asOrderedSet()->at(1)
2527         endif
2528     in
2529     let acs : Set(AssociationClass) = /* associations between them */
2530     AssociationClass.allInstances()->select( ac |
2531         ac.ownedEnd.class = Set {c, myc} )
2532     in
2533     acs->select( ac |
2534         ac.ownedEnd.getAppliedStereotype('CIM::CIM_Reference').
2535         oclAsType(CIM::CIM_Reference).Weak = true)->size() = 1

```

2536 Q50. A CIM class shall not be abstract and have a *Terminal* qualifier with an effective value of true.

2537 Context: Stereotypes *CIM\_Class* and *CIM\_Indication*

2538 OCL: not (self.base\_Class.isAbstract and self.Terminal)

2539 Context: Stereotypes *CIM\_Association*

2540 OCL: not (self.base\_AssociationClass.isAbstract and self.Terminal)

2541 Q51. The number of entries in the *Values* and *ValueMap* qualifier arrays shall match if both are defined.

2542 Context: Stereotype *CIM\_Property*

```

2543 OCL: let s : Stereotype =
2544     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2545     in
2546     self.base_Property.isSet(s, 'ValueMap')
2547     and self.base_Property.isSet(s, 'Values')
2548     implies
2549     self.ValueMap->size() = self.Values->size()

```

2550 Context: Stereotype *CIM\_Method*

```

2551 OCL: let s : Stereotype =
2552     self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
2553     in
2554     self.base_Property.isSet(s, 'ValueMap')
2555     and self.base_Property.isSet(s, 'Values')
2556     implies
2557     self.ValueMap->size() = self.Values->size()

```

2558 Context: Stereotype *CIM\_Parameter*

```

2559 OCL: let s : Stereotype =
2560     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
2561     in
2562     self.base_Property.isSet(s, 'ValueMap')
2563     and self.base_Property.isSet(s, 'Values')
2564     implies

```



2565           self.ValueMap->size() = self.Values->size()

2566 Q52. The *ValueMap* qualifier is applicable only to a CIM string or integer datatype.

2567       Context: Stereotype *CIM\_Property*

2568       OCL: let s : Stereotype =  
 2569           self.base\_Property.getAppliedStereotype('CIM::CIM\_Property')  
 2570       in  
 2571       self.base\_Property.isSet(s, 'ValueMap')  
 2572       implies  
 2573       Set { /\* the integer and string datatypes \*/  
 2574           'CIMDatatypes::uint8',  
 2575           'CIMDatatypes::uint16',  
 2576           'CIMDatatypes::uint32',  
 2577           'CIMDatatypes::uint64',  
 2578           'CIMDatatypes::sint8',  
 2579           'CIMDatatypes::sint16',  
 2580           'CIMDatatypes::sint32',  
 2581           'CIMDatatypes::sint64',  
 2582           'CIMDatatypes::string',  
 2583           'CIMDatatypes::char16'  
 2584       }->includes(self.base\_Property.type.qualifiedName)

2585       Context: Stereotype *CIM\_Method*

2586       OCL: let s : Stereotype =  
 2587           self.base\_Operation.getAppliedStereotype('CIM::CIM\_Method')  
 2588       in  
 2589       self.base\_Operation.isSet(s, 'ValueMap')  
 2590       implies  
 2591       Set { /\* the integer and string datatypes \*/  
 2592           'CIMDatatypes::uint8',  
 2593           'CIMDatatypes::uint16',  
 2594           'CIMDatatypes::uint32',  
 2595           'CIMDatatypes::uint64',  
 2596           'CIMDatatypes::sint8',  
 2597           'CIMDatatypes::sint16',  
 2598           'CIMDatatypes::sint32',  
 2599           'CIMDatatypes::sint64',  
 2600           'CIMDatatypes::string',  
 2601           'CIMDatatypes::char16'  
 2602       }->includes(self.base\_Operation.type.qualifiedName)

2603       Context: Stereotype *CIM\_Parameter*

2604       OCL: let s : Stereotype =  
 2605           self.base\_Parameter.getAppliedStereotype('CIM::CIM\_Parameter')  
 2606       in  
 2607       self.base\_Parameter.isSet(s, 'ValueMap')  
 2608       implies  
 2609       Set { /\* the integer and string datatypes \*/  
 2610           'CIMDatatypes::uint8',  
 2611           'CIMDatatypes::uint16',  
 2612           'CIMDatatypes::uint32',

```

2613         'CIMDatatypes::uint64',
2614         'CIMDatatypes::sint8',
2615         'CIMDatatypes::sint16',
2616         'CIMDatatypes::sint32',
2617         'CIMDatatypes::sint64',
2618         'CIMDatatypes::string',
2619         'CIMDatatypes::char16'
2620     }->includes(self.base_Parameter.type.qualifiedName)

```

2621 Q53. A value of an entry in the *ValueMap* qualifier array when used with an integer datatype shall  
 2622 conform to the format defined in [DSP0004](#).

2623 Represented in ABNF according to 3.4, the format defined in [DSP0004](#) is:

```

2624 ValueMapFormat = [integerValue] "." [integerValue];

```

2625 where the rule on the right hand side of this ABNF rule is defined in Appendix A of [DSP0004](#), and  
 2626 the value range of both occurrences of *integerValue* is constrained by the integer datatype of  
 2627 the CIM element to which this qualifier is applied.

2628 Context: Stereotypes *CIM\_Property*, *CIM\_Method* and *CIM\_Parameter*

```

2629 OCL: self.ValueMap->forall( qv | qv.matchCimAbnf( 'ValueMapFormat' )
2630 )

```

2631 Q54. The value of a *Version* qualifier array shall conform to the format defined in [DSP0004](#).

2632 Represented in ABNF according to 3.4, the format defined in [DSP0004](#) is:

```

2633 VersionFormat = 1*(decimalDigit) "." 1*(decimalDigit)
2634               "." 1*(decimalDigit);

```

2635 where the rule on the right hand side of this ABNF rule is defined in Appendix A of [DSP0004](#).

2636 Context: Stereotypes *CIM\_Class*, *CIM\_Association* and *CIM\_Indication*,

```

2637 OCL: self.Version.matchCimAbnf( 'VersionFormat' )

```

2638 Q55. A CIM association shall have no more than one reference qualified with the *Weak* qualifier.

2639 Context: Stereotype *CIM\_Association*

```

2640 OCL: self.base_AssociationClass.ownedEnd->select( p |
2641         p.isSet(p.getAppliedStereotype( 'CIM::CIM_Reference' ), 'Weak')
2642     )->size() <= 1

```

2643 Q56. The value of a *PropertyUsage* qualifier shall conform to the format defined in [DSP0004](#).

2644 Represented in ABNF according to 3.4, the format defined in [DSP0004](#) is:

```

2645 PropertyUsageFormat = "UNKNOWN" | "OTHER" | "CURRENTCONTEXT" |
2646                     "DESCRIPTIVE" | "CAPABILITY" |
2647                     "CONFIGURATION" | "STATE" | "METRIC";

```

2648 Context: Stereotype *CIM\_Property*

```

2649 OCL: self.PropertyUsage <> null
2650     implies
2651     Bag {
2652         'UNKNOWN', 'OTHER', 'CURRENTCONTEXT',
2653         'DESCRIPTIVE', 'CAPABILITY',
2654         'CONFIGURATION', 'STATE', 'METRIC'

```

2655            }->includes(self.PropertyUsage.toUpper())

2656 Q57. The *Syntax* qualifier and the *SyntaxType* qualifier shall be used together.

2657            Context: Stereotypes *CIM\_Property*, *CIM\_Reference*, *CIM\_Method* and *CIM\_Parameter*

2658            OCL: self.Syntax <> null xor self.SyntaxType = null

2659 Q58. The value of a *TriggerType* qualifier on a CIM non-indication class shall conform to the format

2660 defined in [DSP0004](#).

2661            Represented in ABNF according to 3.4, the format defined in [DSP0004](#) is:

2662            TriggerTypeFormat1 = "CREATE" | "DELETE" | "UPDATE" | "ACCESS";

2663            Context: Stereotype *CIM\_Class*

2664            OCL: self.TriggerType <> null

2665                implies

2666                Bag {

2667                    'CREATE', 'DELETE', 'UPDATE', 'ACCESS'

2668                }->includes(self.TriggerType.toUpper())

2669            Context: Stereotype *CIM\_Association*

2670            OCL: self.TriggerType <> null

2671                implies

2672                Bag {

2673                    'CREATE', 'DELETE', 'UPDATE', 'ACCESS'

2674                }->includes(self.TriggerType.toUpper())

2675 Q59. The value of a *TriggerType* qualifier on a CIM property shall conform to the format defined in

2676 [DSP0004](#).

2677            Represented in ABNF according to 3.4, the format defined in [DSP0004](#) is:

2678            TriggerTypeFormat2 = "UPDATE" | "ACCESS";

2679            Context: Stereotype *CIM\_Property*

2680            OCL: self.TriggerType <> null

2681                implies

2682                Bag {

2683                    'UPDATE', 'ACCESS'

2684                }->includes(self.TriggerType.toUpper())

2685            Context: Stereotype *CIM\_Reference*

2686            OCL: self.TriggerType <> null

2687                implies

2688                Bag {

2689                    'UPDATE', 'ACCESS'

2690                }->includes(self.TriggerType.toUpper())

2691 Q60. The value of a *TriggerType* qualifier on a CIM method shall conform to the format defined in

2692 [DSP0004](#).

2693            Represented in ABNF according to 3.4, the format defined in [DSP0004](#) is:

2694            TriggerTypeFormat2 = "BEFORE" | "AFTER";

2695 Context: Stereotype *CIM\_Method*

2696 OCL: self.TriggerType <> null  
 2697 implies  
 2698 Bag {  
 2699 'BEFORE', 'AFTER'  
 2700 }->includes(self.TriggerType.toUpper())

2701 Q61. The value of a *TriggerType* qualifier on a CIM indication shall conform to the format defined in  
 2702 [DSP0004](#).

2703 Represented in ABNF according to 3.4, the format defined in [DSP0004](#) is:

2704 TriggerTypeFormat3 = "THROWN";

2705 Context: Stereotype *CIM\_Indication*

2706 OCL: self.TriggerType <> null  
 2707 implies  
 2708 Bag {  
 2709 'THROWN'  
 2710 }->includes(self.TriggerType.toUpper())

2711 Q62. A value of an entry in the *UnsupportedValues* qualifier array when used with an integer datatype  
 2712 shall conform to the format defined in [DSP0004](#).

2713 Represented in ABNF according to 3.4, the format defined in [DSP0004](#) is:

2714 UnsupportedValuesFormat = [integerValue] ".." [integerValue];

2715 where the rule on the right hand side of this ABNF rule is defined in Appendix A of [DSP0004](#), and  
 2716 the value range of both occurrences of *integerValue* is constrained by the integer datatype of  
 2717 the CIM element to which this qualifier is applied.

2718 Context: Stereotype *CIM\_Property*

2719 OCL: let s : Stereotype =  
 2720 self.base\_Property.getAppliedStereotype('CIM::CIM\_Property')  
 2721 in  
 2722 self.base\_Property.isSet(s, 'UnsupportedValues')  
 2723 and  
 2724 Set { /\* the integer datatypes \*/  
 2725 'CIMDatatypes::uint8',  
 2726 'CIMDatatypes::uint16',  
 2727 'CIMDatatypes::uint32',  
 2728 'CIMDatatypes::uint64',  
 2729 'CIMDatatypes::sint8',  
 2730 'CIMDatatypes::sint16',  
 2731 'CIMDatatypes::sint32',  
 2732 'CIMDatatypes::sint64'  
 2733 }->includes(self.base\_Property.type.qualifiedName)  
 2734 implies  
 2735 self.UnsupportedValues->forall( qv |  
 2736 qv.matchCimAbnf( ' UnsupportedValuesFormat' )  
 2737 )

2738 Q63. The *Deprecated* qualifier shall not be used with the *Experimental* qualifier

2739 Context: Stereotype *CIM\_Class* and *CIM\_Indication*

2740 OCL: not

```

2741         (self.base_Class.getAppliedStereotype('CIM::Deprecated')
2742             <> null
2743         and self.Experimental
2744     )

```

2745 Context: Stereotype *CIM\_Association*

```

2746 OCL: not (
2747     self.base_AssociationClass.getAppliedStereotype('CIM::Deprecated')
2748     <> null
2749     and self.Experimental
2750 )

```

2751 Context: Stereotypes *CIM\_Property* and *CIM\_Reference*

```

2752 OCL: not (
2753     self.base_Property.getAppliedStereotype('CIM::Deprecated')
2754     <> null
2755     and self.Experimental
2756 )

```

2757 Context: Stereotype *CIM\_Method*

```

2758 OCL: not (
2759     self.base_Operation.getAppliedStereotype('CIM::Deprecated')
2760     <> null
2761     and self.Experimental
2762 )

```

2763 Context: Stereotype *CIM\_Parameter*

```

2764 OCL: not (
2765     self.base_Parameter.getAppliedStereotype('CIM::Deprecated')
2766     <> null
2767     and self.Experimental
2768 )

```

2769 Q64. The *EmbeddedObject* qualifier shall not be used with the *Key* qualifier.

2770 Context: Stereotypes *CIM\_Property*

```

2771 OCL: not (
2772     self.EmbeddedObject and
2773     self.base_Property.getAppliedStereotype('CIM::Key') <> null
2774 )

```

2775 Q65. A property containing an embedded instance by value shall not be qualified with the *Key* qualifier.

2776 Context: Stereotypes *CIM\_Property*

```

2777 OCL: let s : Stereotype =
2778     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
2779     in
2780     not (
2781         self.base_Property.type.ocIsTypeOf(uml::Class) and
2782         self.base_Property.getAppliedStereotype('CIM::Key') <> null
2783     )

```

- 2784 Q66. The name of a CIM qualifier type shall conform to the format for CIM element names defined in  
2785 [DSP0004](#).
- 2786 The format for CIM element names is described in 5.17.1.8.
- 2787 Context: Stereotype *CIM\_QualifierType*
- 2788 OCL: `self.base_Property.name.isValidCimElementName()`
- 2789 Q67. The value of the *ArrayType* stereotype property shall be consistent with the value of the *isOrdered*  
2790 attribute.
- 2791 Context: Stereotype *CIM\_Property*
- 2792 OCL: `if self.base_Property.isOrdered`  
2793 `then`  
2794 `self.ArrayType = CIM::CIM_Qualifier_ArrayType_Enum::Indexed`  
2795 `else`  
2796 `self.ArrayType = CIM::CIM_Qualifier_ArrayType_Enum::Ordered`  
2797 `or`  
2798 `self.ArrayType = CIM::CIM_Qualifier_ArrayType_Enum::Bag`  
2799 `endif`
- 2800 Context: Stereotype *CIM\_Parameter*
- 2801 OCL: `if self.base_Parameter.isOrdered`  
2802 `then`  
2803 `self.ArrayType = CIM::CIM_Qualifier_ArrayType_Enum::Indexed`  
2804 `else`  
2805 `self.ArrayType = CIM::CIM_Qualifier_ArrayType_Enum::Ordered`  
2806 `or`  
2807 `self.ArrayType = CIM::CIM_Qualifier_ArrayType_Enum::Bag`  
2808 `endif`
- 2809 Q68. The value of the *ClassReferenceType* stereotype property of a CIM property of class type shall be  
2810 *ByValue*.
- 2811 Context: Stereotype *CIM\_Property*
- 2812 OCL: `self.base_Property.type.oclIsTypeOf(uml::Class)`  
2813 `implies`  
2814 `self.ClassReferenceType = CIM::CIM_ClassReferenceType_Enum::ByValue`
- 2815 Q69. The value of the *ClassReferenceType* stereotype property of a CIM parameter of class type shall  
2816 be *ByReference* or *ByValue*.
- 2817 Context: Stereotype *CIM\_Parameter*
- 2818 OCL: `self.base_Parameter.type.oclIsTypeOf(uml::Class)`  
2819 `implies (`  
2820 `self.ClassReferenceType = CIM::CIM_ClassReferenceType_Enum::ByValue or`  
2821 `self.ClassReferenceType = CIM::CIM_ClassReferenceType_Enum::ByReference`  
2822 `)`
- 2823 Q70. The value of the *ClassReferenceType* stereotype property of a CIM method with a return value of  
2824 class type shall be *ByReference* or *ByValue*.
- 2825 Context: Stereotype *CIM\_Method*
- 2826 OCL: `let rv : OrderedSet(Parameter) =`  
2827 `self.base_Operation.ownedParameter->select( p |`

```

2828         p.direction = uml::ParameterDirectionKind::return)
2829     in
2830     rv->size() = 1 and
2831     rv->at(1).type.oclIsTypeOf(uml::Class)
2832     implies (
2833         self.ClassReferenceType = CIM::CIM_ClassReferenceType_Enum::ByValue or
2834         self.ClassReferenceType = CIM::CIM_ClassReferenceType_Enum::ByReference
2835     )

```

### 2836 5.17.6 OCL Constraints on Datatypes

2837 T1. The value range of the CIM datatypes uint8, uint16, uint32, and uint64 shall be 0 to  $2^{**N}-1$  where N  
 2838 is 8, 16, 32 and 64, respectively.

2839 Context: A stereotype extending UML *Slot*

2840 OCL: self.base\_Slot.definingFeature.type.qualifiedName =  
 2841 'CIMDatatypes::uint8' implies  
 2842 self.base\_Slot.value.integerValue()->forall( iv | iv >= 0 and iv <= 255  
 2843 )

2844 OCL: self.base\_Slot.definingFeature.type.qualifiedName =  
 2845 'CIMDatatypes::uint16' implies  
 2846 self.base\_Slot.value.integerValue()->forall( iv | iv >= 0 and iv <= 32767  
 2847 )

2848 OCL: self.base\_Slot.definingFeature.type.qualifiedName =  
 2849 'CIMDatatypes::uint32' implies  
 2850 self.base\_Slot.value.integerValue()->forall( iv | iv >= 0 and iv <=  $2^{**32}-1$   
 2851 )

2852 OCL: self.base\_Slot.definingFeature.type.qualifiedName =  
 2853 'CIMDatatypes::uint64' implies  
 2854 self.base\_Slot.value.integerValue()->forall( iv | iv >= 0 and iv <=  $2^{**64}-1$   
 2855 )

2856 T2. The value range of the CIM datatypes sint8, sint16, sint32, and sint64 shall be  $-2^{**N}-1$  to  $2^{**N}-1$  where N is 8, 16, 32 and 64, respectively.

2858 Context: A stereotype extending UML *Slot*

2859 OCL: self.base\_Slot.definingFeature.type.qualifiedName =  
 2860 'CIMDatatypes::sint8' implies  
 2861 self.base\_Slot.value.integerValue()->forall( iv |  
 2862 iv >= -128 and iv <= 127  
 2863 )

2864 OCL: self.base\_Slot.definingFeature.type.qualifiedName =  
 2865 'CIMDatatypes::sint16' implies  
 2866 self.base\_Slot.value.integerValue()->forall( iv |  
 2867 iv >= -32768 and iv <= 32767  
 2868 )

2869 OCL: self.base\_Slot.definingFeature.type.qualifiedName =  
 2870 'CIMDatatypes::sint32' implies  
 2871 self.base\_Slot.value.integerValue()->forall( iv |  
 2872 iv >=  $-2^{**31}$  and iv <=  $2^{**31}-1$

```

2873         )
2874     OCL: self.base_Slot.definingFeature.type.qualifiedName =
2875         'CIMDatatypes::sint64' implies
2876         self.base_Slot.value.integerValue()->forall( iv |
2877             iv >= -2**64 and iv <= 2**64-1
2878         )

2879 T3. The value range of the CIM datatypes real32 and real64 shall be the value range specified for IEEE
2880 4-byte and 8-byte floating point values, respectively, as defined in IEEE Standard 754.
2881 Context: A stereotype extending UML Slot
2882 OCL: self.base_Slot.definingFeature.type.qualifiedName =
2883     'CIMDatatypes::real32' implies
2884     self.base_Slot.value.integerValue()->forall( iv | iv.isValidCimReal32()
2885     )

2886 OCL: self.base_Slot.definingFeature.type.qualifiedName =
2887     'CIMDatatypes::real64' implies
2888     self.base_Slot.value.integerValue()->forall( iv | iv.isValidCimReal64()
2889     )

2890 T4. The value range of the CIM datatype char16 shall be one Unicode character from the UCS-2
2891 range.
2892 Context: A stereotype extending UML Slot
2893 OCL: self.base_Slot.definingFeature.type.qualifiedName =
2894     'CIMDatatypes::char16' implies
2895     self.base_Slot.value.stringValue()->forall( sv | sv.size() <= 1 and
2896         sv.isValidCimString()
2897     )

2898 T5. The value range of the CIM datatype string shall be an ordered array of Unicode characters from
2899 the UCS-2 range.
2900 Context: A stereotype extending UML Slot
2901 OCL: self.base_Slot.definingFeature.type.qualifiedName =
2902     'CIMDatatypes::string' implies
2903     self.base_Slot.value.stringValue()->forall( sv | sv.isValidCimString()
2904     )

2905 T6. The value range of the CIM datatype datetime shall be a string of 24 Unicode characters as defined
2906 in DSP0004.
2907 Context: A stereotype extending UML Slot
2908 OCL: self.base_Slot.definingFeature.type.qualifiedName =
2909     'CIMDatatypes::datetime' implies
2910     self.base_Slot.value.stringValue()->forall( sv | sv.isValidCimDatetime()
2911     )

2912 T7. The format of a value of a CIM::octetstring datatype shall conform to the format defined in
2913 DSP0004.
2914 Represented in ABNF according to 3.4, the format defined in DSP0004 is:
2915 OctetstringFormat = "0x" 4*(hexDigit hexDigit);

```



2916 where the rule on the right hand side of this ABNF rule is defined in Appendix A of [DSP0004](#), and  
 2917 the first 4 octets of the octet string (8 hexadecimal digits in the text encoding) are the number of  
 2918 octets in the represented octet string in big endian format with the length portion included in the  
 2919 octet count.

2920 NOTE: This rule covers octetstrings with both variants of CIM datatypes, array of string and array of uint8.

2921 Context: A stereotype extending UML *Slot*

```
2922 OCL: self.base_Slot.definingFeature.type.qualifiedName =
2923       'CIMDatatypes::octetstring' implies
2924       self.base_Slot.value.stringValue()->forall( sv |
2925         sv.matchCimAbnf( 'OctetstringFormat' ) and
2926         sv.size() = sv.substring(3,10).hexToInteger() * 2 + 2
2927       )
```

### 2928 5.17.7 OCL Constraints for Values of Directly Mapped Qualifiers

2929 The OCL constraints in this subclause verify that directly mapped qualifiers do not have a value in the  
 2930 corresponding property of their qualifier type stereotype.

2931 NOTE: The *ArrayType* qualifier is mapped to the stereotype property in addition to UML constructs, therefore it is not  
 2932 included in this subclause.

2933 These OCL constraints should be surfaced with less severity than the others.

2934 W1. Stereotype property *Abstract* shall not have a value – use the UML *isAbstract* attribute instead

2935 Context: Stereotype *CIM\_Class*

```
2936 OCL: let s : Stereotype = self.base_Class.getAppliedStereotype('CIM::CIM_Class')
2937     in
2938     self.base_Class.isSet(s,'Abstract') = false
```

2939 Context: Stereotype *CIM\_Association*

```
2940 OCL: let s : Stereotype =
2941       self.base_AssociationClass.getAppliedStereotype('CIM::CIM_Association')
2942     in
2943     self.base_AssociationClass.isSet(s,'Abstract') = false
```

2944 Context: Stereotype *CIM\_Indication*

```
2945 OCL: let s : Stereotype =
2946       self.base_Class.getAppliedStereotype('CIM::CIM_Indication')
2947     in
2948     self.base_Class.isSet(s,'Abstract') = false
```

2949 W2. Stereotype property *Association* shall not have a value – already indicated by using UML  
 2950 *AssociationClass*.

2951 Context: Stereotype *CIM\_Association*

```
2952 OCL: let s : Stereotype =
2953       self.base_AssociationClass.getAppliedStereotype('CIM::CIM_Association')
2954     in
2955     self.base_AssociationClass.isSet(s,'Association') = false
```

- 2956 W3. Stereotype property *Aggregation* shall not have a value – use the UML aggregation indicator  
2957 instead.
- 2958 Context: Stereotype *CIM\_Association*
- 2959 OCL: let s : Stereotype =  
2960 self.base\_AssociationClass.getAppliedStereotype('CIM::CIM\_Association')  
2961 in  
2962 self.base\_AssociationClass.isSet(s,'Aggregation') = false
- 2963 W4. Stereotype property *Composition* shall not have a value – use the UML aggregation indicator  
2964 instead.
- 2965 Context: Stereotype *CIM\_Association*
- 2966 OCL: let s : Stereotype =  
2967 self.base\_AssociationClass.getAppliedStereotype('CIM::CIM\_Association')  
2968 in  
2969 self.base\_AssociationClass.isSet(s,'Composition') = false
- 2970 W5. Stereotype property *Aggregate* shall not have a value – use the UML aggregation indicator instead.
- 2971 Context: Stereotype *CIM\_Reference*
- 2972 OCL: let s : Stereotype =  
2973 self.base\_Property.getAppliedStereotype('CIM::CIM\_Reference')  
2974 in  
2975 self.base\_Property.isSet(s,'Aggregate') = false
- 2976 W6. Stereotype property *Description* shall not have a value – use the documentation field instead.
- 2977 Context: Stereotype *CIM\_Class*
- 2978 OCL: let s : Stereotype =  
2979 self.base\_Class.getAppliedStereotype('CIM::CIM\_Class')  
2980 in  
2981 self.base\_Class.isSet(s,'Description') = false
- 2982 Context: Stereotype *CIM\_Association*
- 2983 OCL: let s : Stereotype =  
2984 self.base\_AssociationClass.getAppliedStereotype('CIM::CIM\_Association')  
2985 in  
2986 self.base\_AssociationClass.isSet(s,'Description') = false
- 2987 Context: Stereotype *CIM\_Indication*
- 2988 OCL: let s : Stereotype =  
2989 self.base\_Class.getAppliedStereotype('CIM::CIM\_Indication')  
2990 in  
2991 self.base\_Class.isSet(s,'Description') = false
- 2992 Context: Stereotype *CIM\_Property*
- 2993 OCL: let s : Stereotype =  
2994 self.base\_Property.getAppliedStereotype('CIM::CIM\_Property')  
2995 in  
2996 self.base\_Property.isSet(s,'Description') = false
- 2997 Context: Stereotype *CIM\_Reference*
- 2998 OCL: let s : Stereotype =  
2999 self.base\_Property.getAppliedStereotype('CIM::CIM\_Reference')  
3000 in

```

3001         self.base_Property.isSet(s,'Description') = false
3002     Context: Stereotype CIM_Method
3003     OCL: let s : Stereotype =
3004         self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
3005     in
3006         self.base_Operation.isSet(s,'Description') = false

3007     Context: Stereotype CIM_Parameter
3008     OCL: let s : Stereotype =
3009         self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
3010     in
3011         self.base_Parameter.isSet(s,'Description') = false

3012 W7. Stereotype property Experimental shall not have a value – use the marker stereotype Experimental
3013 instead.

3014     Context: Stereotype CIM_Class
3015     OCL: let s : Stereotype = self.base_Class.getAppliedStereotype('CIM::CIM_Class')
3016     in
3017         self.base_Class.isSet(s,'Experimental') = false

3018     Context: Stereotype CIM_Association
3019     OCL: let s : Stereotype =
3020         self.base_AssociationClass.getAppliedStereotype('CIM::CIM_Association')
3021     in
3022         self.base_AssociationClass.isSet(s,'Experimental') = false

3023     Context: Stereotype CIM_Indication
3024     OCL: let s : Stereotype =
3025         self.base_Class.getAppliedStereotype('CIM::CIM_Indication')
3026     in
3027         self.base_Class.isSet(s,'Experimental') = false

3028     Context: Stereotype CIM_Property
3029     OCL: let s : Stereotype =
3030         self.base_Property.getAppliedStereotype('CIM::CIM_Property')
3031     in
3032         self.base_Property.isSet(s,'Experimental') = false

3033     Context: Stereotype CIM_Reference
3034     OCL: let s : Stereotype =
3035         self.base_Property.getAppliedStereotype('CIM::CIM_Reference')
3036     in
3037         self.base_Property.isSet(s,'Experimental') = false

3038     Context: Stereotype CIM_Method
3039     OCL: let s : Stereotype =
3040         self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
3041     in
3042         self.base_Operation.isSet(s,'Experimental') = false

3043     Context: Stereotype CIM_Parameter
3044     OCL: let s : Stereotype =

```

```

3045         self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
3046     in
3047         self.base_Parameter.isSet(s,'Experimental') = false

```

3048 W8. Stereotype property *In* shall not have a value – use the UML direction indicator instead.

3049 Context: Stereotype *CIM\_Parameter*

```

3050 OCL: let s : Stereotype =
3051     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
3052 in
3053     self.base_Parameter.isSet(s,'In') = false

```

3054 W9. Stereotype property *Out* shall not have a value – use the UML direction indicator instead.

3055 Context: Stereotype *CIM\_Parameter*

```

3056 OCL: let s : Stereotype =
3057     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
3058 in
3059     self.base_Parameter.isSet(s,'Out') = false

```

3060 W10. Stereotype property *Key* shall not have a value – use the marker stereotype *Key* instead.

3061 Context: Stereotype *CIM\_Property*

```

3062 OCL: let s : Stereotype =
3063     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
3064 in
3065     self.base_Property.isSet(s,'Key') = false

```

3066 Context: Stereotype *CIM\_Reference*

```

3067 OCL: let s : Stereotype =
3068     self.base_Property.getAppliedStereotype('CIM::CIM_Reference')
3069 in
3070     self.base_Property.isSet(s,'Key') = false

```

3071 W11. Stereotype property *Min* shall not have a value – use the UML association multiplicity instead.

3072 Context: Stereotype *CIM\_Reference*

```

3073 OCL: let s : Stereotype =
3074     self.base_Property.getAppliedStereotype('CIM::CIM_Reference')
3075 in
3076     self.base_Property.isSet(s,'Min') = false

```

3077 W12. Stereotype property *Max* shall not have a value – use the UML association multiplicity instead.

3078 Context: Stereotype *CIM\_Reference*

```

3079 OCL: let s : Stereotype =
3080     self.base_Property.getAppliedStereotype('CIM::CIM_Reference')
3081 in
3082     self.base_Property.isSet(s,'Max') = false

```

3083 W13. Stereotype property *ClassConstraint* shall not have a value – use UML user model constraints instead.

3084

3085 Context: Stereotype *CIM\_Class*

```

3086 OCL: let s : Stereotype = self.base_Class.getAppliedStereotype('CIM::CIM_Class')
3087 in
3088     self.base_Class.isSet(s,'ClassConstraint') = false

```

3089 Context: Stereotype *CIM\_Association*  
 3090 OCL: let s : Stereotype =  
 3091 self.base\_AssociationClass.getAppliedStereotype('CIM::CIM\_Association')  
 3092 in  
 3093 self.base\_AssociationClass.isSet(s,'ClassConstraint') = false

3094 Context: Stereotype *CIM\_Indication*  
 3095 OCL: let s : Stereotype =  
 3096 self.base\_Class.getAppliedStereotype('CIM::CIM\_Indication')  
 3097 in  
 3098 self.base\_Class.isSet(s,'ClassConstraint') = false

3099 W14. Stereotype property *MethodConstraint* shall not have a value – use UML user model constraints  
 3100 instead.

3101 Context: Stereotype *CIM\_Method*  
 3102 OCL: let s : Stereotype =  
 3103 self.base\_Operation.getAppliedStereotype('CIM::CIM\_Method')  
 3104 in  
 3105 self.base\_Operation.isSet(s,'MethodConstraint') = false

3106 W15. Stereotype property *PropertyConstraint* shall not have a value – use UML user model constraints  
 3107 instead.

3108 Context: Stereotype *CIM\_Property*  
 3109 OCL: let s : Stereotype =  
 3110 self.base\_Property.getAppliedStereotype('CIM::CIM\_Property')  
 3111 in  
 3112 self.base\_Property.isSet(s,'PropertyConstraint') = false

3113 Context: Stereotype *CIM\_Reference*  
 3114 OCL: let s : Stereotype =  
 3115 self.base\_Property.getAppliedStereotype('CIM::CIM\_Reference')  
 3116 in  
 3117 self.base\_Property.isSet(s,'PropertyConstraint') = false

3118 W16. Stereotype property *Octetstring* shall not have a value – use the type CIMDatatypes::octetstring  
 3119 instead.

3120 Context: Stereotype *CIM\_Method*  
 3121 OCL: let s : Stereotype =  
 3122 self.base\_Operation.getAppliedStereotype('CIM::CIM\_Method')  
 3123 in  
 3124 self.base\_Operation.isSet(s,'Octetstring') = false

3125 Context: Stereotype *CIM\_Property*  
 3126 OCL: let s : Stereotype =  
 3127 self.base\_Property.getAppliedStereotype('CIM::CIM\_Property')  
 3128 in  
 3129 self.base\_Property.isSet(s,'Octetstring') = false

3130 Context: Stereotype *CIM\_Parameter*  
 3131 OCL: let s : Stereotype =  
 3132 self.base\_Parameter.getAppliedStereotype('CIM::CIM\_Parameter')  
 3133 in

3134           self.base\_Parameter.isSet(s,'Octetstring') = false

3135 W17. Stereotype property *Override* shall not have a value – use the UML redefinition capabilities instead.

3136       Context: Stereotype *CIM\_Method*

3137       OCL: let s : Stereotype =

3138           self.base\_Operation.getAppliedStereotype('CIM::CIM\_Method')

3139       in

3140       self.base\_Operation.isSet(s,'Override') = false

3141       Context: Stereotype *CIM\_Property*

3142       OCL: let s : Stereotype =

3143           self.base\_Property.getAppliedStereotype('CIM::CIM\_Property')

3144       in

3145       self.base\_Property.isSet(s,'Override') = false

3146       Context: Stereotype *CIM\_Reference*

3147       OCL: let s : Stereotype =

3148           self.base\_Property.getAppliedStereotype('CIM::CIM\_Reference')

3149       in

3150       self.base\_Property.isSet(s,'Override') = false

3151 W18. Stereotype property *Static* shall not have a value – use the UML static indicator instead.

3152       Context: Stereotype *CIM\_Method*

3153       OCL: let s : Stereotype =

3154           self.base\_Operation.getAppliedStereotype('CIM::CIM\_Method')

3155       in

3156       self.base\_Operation.isSet(s,'Static') = false

3157       Context: Stereotype *CIM\_Property*

3158       OCL: let s : Stereotype =

3159           self.base\_Property.getAppliedStereotype('CIM::CIM\_Property')

3160       in

3161       self.base\_Property.isSet(s,'Static') = false

3162 W19. Stereotype property *Terminal* shall not have a value – use the UML leaf indicator instead.

3163       Context: Stereotype *CIM\_Class*

3164       OCL: let s : Stereotype = self.base\_Class.getAppliedStereotype('CIM::CIM\_Class')

3165       in

3166       self.base\_Class.isSet(s,'Terminal') = false

3167       Context: Stereotype *CIM\_Association*

3168       OCL: let s : Stereotype =

3169           self.base\_AssociationClass.getAppliedStereotype('CIM::CIM\_Association')

3170       in

3171       self.base\_AssociationClass.isSet(s,'Terminal') = false

3172       Context: Stereotype *CIM\_Indication*

3173       OCL: let s : Stereotype =

3174           self.base\_Class.getAppliedStereotype('CIM::CIM\_Indication')

3175       in

3176       self.base\_Class.isSet(s,'Terminal') = false

3177 W20. Stereotype property *Write* shall not have a value – use the UML read-only indicator instead.

3178 Context: Stereotype *CIM\_Property*

```
3179 OCL: let s : Stereotype =
3180     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
3181     in
3182     self.base_Property.isSet(s,'Write') = false
```

3183 W21. Stereotype property *EmbeddedInstance* shall not have a value – use a class as a type instead.

3184 Context: Stereotype *CIM\_Property*

```
3185 OCL: let s : Stereotype =
3186     self.base_Property.getAppliedStereotype('CIM::CIM_Property')
3187     in
3188     self.base_Property.isSet(s,'EmbeddedInstance') = false
```

3189 Context: Stereotype *CIM\_Method*

```
3190 OCL: let s : Stereotype =
3191     self.base_Operation.getAppliedStereotype('CIM::CIM_Method')
3192     in
3193     self.base_Operation.isSet(s,'EmbeddedInstance') = false
```

3194 Context: Stereotype *CIM\_Parameter*

```
3195 OCL: let s : Stereotype =
3196     self.base_Parameter.getAppliedStereotype('CIM::CIM_Parameter')
3197     in
3198     self.base_Parameter.isSet(s,'EmbeddedInstance') = false
```

### 3199 5.17.8 Other OCL Constraints

3200 O1. A comment shall not have further comments.

3201 Context: A stereotype extending UML *Comment*

```
3202 OCL: self.base_Comment.ownedComment->isEmpty()
```

3203 O2. A CIM constraint shall have public visibility.

3204 Context: A stereotype extending UML *Constraint*

```
3205 OCL: self.base_Constraint.visibility = uml::VisibilityKind::public
```

3206 O3. A CIM constraint shall not have a name.

3207 Context: A stereotype extending UML *Constraint*

```
3208 OCL: self.base_Constraint.name = null
3209     or
3210     self.base_Constraint.name = ''
```

3211 O4. A CIM constraint shall not have a comment.

3212 Context: A stereotype extending UML *Constraint*

```
3213 OCL: self.base_Constraint.ownedComment->isEmpty()
```

3214 O5. The constraint expression of a CIM constraint shall have public visibility.

3215 Context: A stereotype extending UML *Constraint*

```
3216 OCL: self.base_Constraint.specification.visibility = uml::VisibilityKind::public
```

- 3217 O6. The constraint expression of a CIM constraint shall not have a name.  
 3218 Context: A stereotype extending UML *Constraint*  
 3219 OCL: `self.base_Constraint.specification.name = null`  
 3220 `or`  
 3221 `self.base_Constraint.specification.name = ''`
- 3222 O7. The constraint expression of a CIM constraint shall not have a comment.  
 3223 Context: A stereotype extending UML *Constraint*  
 3224 OCL: `self.base_Constraint.specification.ownedComment->isEmpty()`
- 3225 O8. The constraint expression of a CIM constraint shall use OCL as a language.  
 3226 Context: A stereotype extending UML *Constraint*  
 3227 OCL: `let lang : OrderedSet(String) = self.base_Constraint.specification.`  
 3228 `oclAsType(uml::OpaqueExpression).language`  
 3229 `in`  
 3230 `lang->isEmpty() /* OCL is implied if no language specified */`  
 3231 `or`  
 3232 `( lang->size() = 1 and lang->at(1) = 'OCL' )`
- 3233 O9. A *LiteralSpecification* shall not have a name.  
 3234 Context: A stereotype extending UML *LiteralSpecification*  
 3235 OCL: `self.base_LiteralSpecification.name = null`  
 3236 `or`  
 3237 `self.base_LiteralSpecification.name = ''`
- 3238 O10. A *LiteralSpecification* shall have public visibility.  
 3239 Context: A stereotype extending UML *LiteralSpecification*  
 3240 OCL: `self.base_LiteralSpecification.visibility = uml::VisibilityKind::public`
- 3241 O11. A *LiteralSpecification* shall not reference a type.  
 3242 Context: A stereotype extending UML *LiteralSpecification*  
 3243 OCL: `self.base_LiteralSpecification.type->isEmpty()`
- 3244 O12. A *LiteralSpecification* shall not have a comment.  
 3245 Context: A stereotype extending UML *LiteralSpecification*  
 3246 OCL: `self.base_LiteralSpecification.ownedComment->isEmpty()`
- 3247 O13. An *Enumeration* shall not be abstract.  
 3248 Context: Stereotype *CIM\_Enumeration*  
 3249 OCL: `self.base_Enumeration.isAbstract = false`
- 3250 O14. An *Enumeration* shall not be a leaf.  
 3251 Context: Stereotype *CIM\_Enumeration*  
 3252 OCL: `self.base_Enumeration.isLeaf = false`



- 3253 O15. An *Enumeration* shall be defined nested in a *Class*.
- 3254 Context: Stereotype *CIM\_Enumeration*
- 3255 OCL: `self.base_Enumeration.owner.oclIsTypeOf(uml::Class)`
- 3256 O16. An *Enumeration* shall be used as a type for a property, operation parameter, or operation return  
3257 value defined in the *Class* defining the *Enumeration*.
- 3258 Context: Stereotype *CIM\_Enumeration*
- 3259 OCL: `let enumParms : Sequence(Parameter) =`  
3260 `self.base_Enumeration.owner.oclAsType(uml::Class)`  
3261 `.ownedOperation.ownedParameter`  
3262 `->select( p | p.type = self.base_Enumeration)`  
3263 `in`  
3264 `let enumProps : OrderedSet(Property) =`  
3265 `self.base_Enumeration.owner.oclAsType(uml::Class).ownedAttribute`  
3266 `->select( p | p.type = self.base_Enumeration)`  
3267 `in`  
3268 `enumProps->size() = 1 xor`  
3269 `enumParms->size() = 1`
- 3270 O17. An *Enumeration* shall have public visibility.
- 3271 Context: Stereotype *CIM\_Enumeration*
- 3272 OCL: `self.base_Enumeration.visibility = uml::VisibilityKind::public`
- 3273 O18. An *Enumeration* shall not own UML attributes.
- 3274 Context: Stereotype *CIM\_Enumeration*
- 3275 OCL: `self.base_Enumeration.ownedAttribute->isEmpty()`
- 3276 O19. An *Enumeration* shall not own UML operations.
- 3277 Context: Stereotype *CIM\_Enumeration*
- 3278 OCL: `self.base_Enumeration.ownedOperation->isEmpty()`
- 3279 O20. An *Enumeration* shall not have superclasses other than either one CIM datatype or one other  
3280 enumeration.
- 3281 Context: Stereotype *CIM\_Enumeration*
- 3282 OCL: `self.base_Enumeration.generalization->isEmpty() = false`  
3283 `implies`  
3284 `self.base_Enumeration.generalization->size() = 1 and (`  
3285 `self.base_Enumeration.generalization->asOrderedSet()`  
3286 `->at(1).general.namespace.name = 'CIMDatatypes' or`  
3287 `self.base_Enumeration.generalization->asOrderedSet()`  
3288 `->at(1).general.oclIsTypeOf(uml::Enumeration)`  
3289 `)`
- 3290 O21. An *Enumeration* shall not redefine another classifier.
- 3291 Context: Stereotype *CIM\_Enumeration*
- 3292 OCL: `self.base_Enumeration.redefinedClassifier->isEmpty()`

- 3293 O22. An *Enumeration* shall not import any elements.  
 3294 Context: Stereotype *CIM\_Enumeration*  
 3295 OCL: `self.base_Enumeration.elementImport->isEmpty()`
- 3296 O23. An *Enumeration* shall not import any packages.  
 3297 Context: Stereotype *CIM\_Enumeration*  
 3298 OCL: `self.base_Enumeration.packageImport->isEmpty()`
- 3299 O24. An *Enumeration* shall not own any UML comments.  
 3300 Context: Stereotype *CIM\_Enumeration*  
 3301 OCL: `self.base_Enumeration.ownedComment->isEmpty()`
- 3302 O25. An *Enumeration* shall have its *IsValuesDefined* or its *IsValueMapDefined* stereotype property set to  
 3303 true.  
 3304 Context: Stereotype *CIM\_Enumeration*  
 3305 OCL: `self.IsValuesDefined or self.IsValueMapDefined`
- 3306 O26. An Enumeration used as a type for a CIM property shall have the name: <property-name> '\_Enum'  
 3307 Context: Stereotype *CIM\_Property*  
 3308 OCL: `self.base_Property.type.oclIsTypeOf(uml::Enumeration)`  
 3309 `implies`  
 3310 `self.base_Property.type.name =`  
 3311 `self.base_Property.name`  
 3312 `.concat('_Enum')`
- 3313 O27. An Enumeration used as a type for a CIM method return value shall have the name: <method-  
 3314 name> '\_Enum'  
 3315 Context: A stereotype extending UML *Parameter*  
 3316 OCL: `self.base_Parameter.type.oclIsTypeOf(uml::Enumeration) and`  
 3317 `self.base_Parameter.direction = uml::ParameterDirectionKind::return`  
 3318 `implies`  
 3319 `self.base_Parameter.type.name =`  
 3320 `self.base_Parameter.operation.name`  
 3321 `.concat('_Enum')`
- 3322 O28. An Enumeration used as a type for a CIM parameter shall have the name: <method-name> '\_'  
 3323 <parameter-name> '\_Enum'  
 3324 Context: Stereotype *CIM\_Parameter*  
 3325 OCL: `self.base_Parameter.type.oclIsTypeOf(uml::Enumeration) and`  
 3326 `self.base_Parameter.direction <>uml::ParameterDirectionKind::return`  
 3327 `implies`  
 3328 `self.base_Parameter.type.name =`  
 3329 `self.base_Parameter.operation.name`  
 3330 `.concat('_')`  
 3331 `.concat(self.base_Parameter.name)`  
 3332 `.concat('_Enum')`

3333 O29. An *EnumerationLiteral* in an *Enumeration* with its *IsValuesDefined* stereotype property set to false  
 3334 shall have a value that equals its name.

3335 Context: A stereotype extending UML *EnumerationLiteral*

```
3336 OCL: self.base_EnumerationLiteral.enumeration
3337     .getAppliedStereotype('CIM::CIM_Enumeration')
3338     .oclAsType(CIM::CIM_Enumeration).IsValuesDefined = false
3339 implies (
3340     let s : ValueSpecification =
3341         self.base_EnumerationLiteral.specification
3342     in
3343     if s.oclIsTypeOf(uml::LiteralString) then
3344         self.base_EnumerationLiteral.name =
3345             s.oclAsType(uml::LiteralString).value
3346     else
3347         self.base_EnumerationLiteral.name =
3348             s.oclAsType(uml::LiteralInteger).value
3349     endif
3350 )
```

3351 O30. An *EnumerationLiteral* in an *Enumeration* with its *IsValueMapDefined* stereotype property set to  
 3352 false shall have a value that equals its position in the *Enumeration*.

3353 Context: A stereotype extending UML *EnumerationLiteral*

```
3354 OCL: self.base_EnumerationLiteral.enumeration
3355     .getAppliedStereotype('CIM::CIM_Enumeration')
3356     .oclAsType(CIM::CIM_Enumeration).IsValueMapDefined = false
3357 implies (
3358     let i : Integer =
3359         self.base_EnumerationLiteral.specification
3360         .oclAsType(uml::LiteralInteger).value
3361     in
3362     self.base_EnumerationLiteral.enumeration
3363     .ownedLiteral->asOrderedSet()->at(i)
3364     = self.base_EnumerationLiteral
3365 )
```

3366 O31. An *EnumerationLiteral* shall have public visibility.

3367 Context: A stereotype extending UML *EnumerationLiteral*

3368 OCL: self.base\_EnumerationLiteral.visibility = uml::VisibilityKind::public

3369 O32. An *EnumerationLiteral* shall not have UML slots.

3370 Context: A stereotype extending UML *EnumerationLiteral*

3371 OCL: self.base\_EnumerationLiteral.slot->isEmpty()

3372 O33. An *EnumerationLiteral* shall have one *LiteralInteger* or *LiteralString* as a value.

3373 Context: A stereotype extending UML *EnumerationLiteral*

```
3374 OCL: let s : ValueSpecification = self.base_EnumerationLiteral.specification
3375     in
3376     s->size() = 1 and (
3377         s.oclIsTypeOf(uml::LiteralInteger) or
3378         s.oclIsTypeOf(uml::LiteralString)
```

3379                    )

### 3380 **5.17.9 Other Constraints**

3381 This subclause lists constraints that cannot be expressed in OCL.

3382 N1. A CIM qualifier with inheritance flavor *ToSubclass DisableOverride* shall not be overridden.

### 3383 **5.17.10 Constraints Covered by UML**

3384 This subclause lists CIM constraints for which the CIM related profiles do not need to define specific OCL  
3385 constraints because these situations are handled by constraints defined by UML, and so a UML tool  
3386 should be supporting them already.

3387 U1. A CIM property may be owned by only one CIM class.

3388 U2. A CIM method may be owned by only one CIM class.

3389 U3. The CIM class overriding a CIM property shall be a subclass of the class defining the property.

3390 U4. The CIM class overriding a CIM method shall be a subclass of the class defining the method.

3391 U5. The name of a property defined in a CIM class shall be unique within the defining class.

3392 U6. The name of a method defined in a CIM class shall be unique within the defining class.

3393 U7. The name of a CIM method parameter shall be unique within the defining method.

3394 U8. A CIM association that inherits another association shall have the same arity.

3395 U9. The arity of a CIM association shall be at least two.

3396 U10. The default value of a CIM property shall match the type of the property.

3397 U11. The name of a CIM qualifier type shall be unique within the owning profile.

3398 U12. A CIM reference shall be owned by a CIM association.

3399 U13. A CIM reference shall not be an array.

3400 U14. The CIM element referenced in an Override qualifier value shall exist in one of the superclasses.

3401 U15. The constraint expression of a CIM OCL constraint shall be of type UML Boolean.

3402 U16. The Min qualifier shall not be NULL.

3403 U17. The Octetstring qualifier is applicable only to CIM string array and uint8 array datatypes.

### 3404 **5.18 Extension Points**

3405 This subclause defines extension points that may be used by conforming implementations of the UML  
3406 profile for CIM in order to support additional functionality.

#### 3407 **5.18.1 Attributes and Associations in UML Metaclasses Used for the Mapping**

3408 The UML metaclasses used in the mapping define several extension points at the level of attributes or  
3409 associations of such metaclasses. In the mapping tables in this document, they are all marked with the  
3410 text "extension point". These extension points are further defined here:

- 3411 • *name* attribute in *InstanceSpecification* metaclass used for modeled CIM instances. This allows  
3412 defining a name for modeled CIM instances in UML although this is not supported in CIM MOF.
- 3413 • *ownedComment* association in *InstanceSpecification* metaclass used for modeled CIM  
3414 instances. This allows defining comments or descriptions on modeled CIM instances. Note that  
3415 such comments or descriptions will not be represented in CIM MOF.
- 3416 • *ownedComment* association in *Slot* metaclass used for property values of modeled CIM  
3417 instances. This allows defining comments or descriptions on property values of modeled CIM  
3418 instances. Note that such comments or descriptions will not be represented in CIM MOF.
- 3419 • *icon* association in *Stereotype* metaclass used as Qualifier Type Stereotype. This allows usage  
3420 of a specific icon for qualifier types. Note that such icons will not be represented in CIM MOF.
- 3421 • *icon* association in *Stereotype* metaclass used as Meta Element Stereotypes. This allows usage  
3422 of a specific icon for each CIM meta element. Note that such icons will not be represented in  
3423 CIM MOF.
- 3424 • *ownedComment* association in *Property* metaclass used for definitional aspects of CIM qualifier  
3425 value within Qualifier Type Stereotype. This allows defining comments or descriptions on  
3426 qualifier types. Note that such comments or descriptions will not be represented in CIM MOF.
- 3427 • *name* attribute in *Constraint* metaclass used for OCL related constraint qualifiers. This allows  
3428 giving these OCL constraints a name to distinguish them. Note that such names will not be  
3429 represented in CIM MOF.
- 3430 • *ownedRule* association in *Enumeration* metaclass used for used for *Values* and *ValueMap*  
3431 qualifiers. This allows defining additional constraints on such enumerations. Note that such  
3432 additional constraints will not be represented in CIM MOF.
- 3433 • *ownedComment* association in *EnumerationLiteral* metaclass used for pair of *Values* and  
3434 *ValueMap* qualifier values. This allows defining comments or descriptions on each pair of  
3435 *Values* and *ValueMap* qualifier values. Note that such comments or descriptions will not be  
3436 represented in CIM MOF.
- 3437 • *ownedRule* association in *DataType* metaclass used for CIM datatypes. This allows defining  
3438 additional constraints on such datatypes (for example value ranges or patterns). Note that such  
3439 additional constraints will not be represented in CIM MOF.

### 3440 5.18.2 Additional Metamodel Constraints

3441 Conforming implementations of the UML profile for CIM may define additional constraints in the UML  
3442 profiles "CIM" and "CIMQualifierType" and in the UML type library "CIMDatatypes", as long as such  
3443 constraints are supported by [DSP0004](#).

### 3444 5.18.3 Additional Stereotypes

3445 Conforming implementations of the UML profile for CIM may define additional stereotypes on any UML  
3446 metaclasses in the UML profiles "CIM" and "CIMQualifierType". Those stereotypes may own properties.

3447 Conforming implementations of the UML profile for CIM may define additional properties on any  
3448 stereotypes defined in this document.

3449 If such property values cannot be represented in CIM MOF, their values will be lost when exporting the  
3450 UML model to CIM MOF.

## 3451 5.19 Compatibility Considerations

3452 There are two kinds of compatibility of future versions of this document:

- 3453 • Compatibility with respect to the UML profile for CIM

3454 A future release of this document is said to be downward compatible with respect to the UML  
 3455 profile for CIM if it does not mandate any changes in the definition of the UML profiles and type  
 3456 libraries defined in 5.21.

- 3457 • Compatibility with respect to user models using the UML profile for CIM

3458 A future release of this document is said to be downward compatible with respect to user  
 3459 models using the UML profile for CIM if it does not mandate any changes in the definition of the  
 3460 user model, except for possibly referencing updated versions of the UML profiles and type  
 3461 libraries defined in 5.21.

3462 Any versions of this document that have the same major version number shall be compatible to each  
 3463 other with respect to user models using the UML profile for CIM and should be compatible to each other  
 3464 with respect to the UML profile for CIM.

3465 This definition of compatibility allows minor version updates of this document to allow elements to be  
 3466 associated with UML metaclass association ends that currently are required to have no elements  
 3467 associated. For example, the *packageImport* association of the UML metaclass *Class* — which currently  
 3468 is required to have no associated elements — might be used for some purpose in the future.

3469 This may cause the UML profiles and type libraries for CIM to be updated in order to support future minor  
 3470 version updates of this document, and in order to deal with user models being transferred from other UML  
 3471 tools that already have updated their UML profiles and type libraries.

## 3472 5.20 Limitations

3473 This subclause lists any limitations of the mapping of CIM to UML, from a perspective of CIM.

- 3474 • The arity of aggregations and compositions is limited to two, while CIM allows their arity to be  
 3475 larger than two. Note that for associations that are not aggregations or compositions, any arity is  
 3476 supported.

3477 Reason: UML 2 restricts the arity of aggregations and compositions to two.

- 3478 • The definition of CIM flavors on qualifier values is not supported. Note that CIM flavors used as  
 3479 part of qualifier type declarations are supported.

3480 Reason: Flavors on qualifier values are intended to be removed from CIM, are not used in the  
 3481 current CIM Schema, and would have introduced additional complexity.

- 3482 • Information about MOF files is mapped in a limited way.

3483 Any `#pragma` directives are represented only from the point of view of their values relevant for  
 3484 CIM classes and CIM instances. This allows reconstructing semantically equivalent CIM MOF  
 3485 files from the UML user model, but information such as where exactly a `#pragma` was defined in  
 3486 the original CIM MOF file will not be represented in the UML user model. Also, the include  
 3487 hierarchy of CIM MOF files is not represented in the UML user model.

- 3488 • CIM elements qualified with *EmbeddedObject* are treated like ordinary strings instead of  
 3489 mapping their types to classes.

3490 Reason: It is not known in the user model whether the CIM element represents an instance or a  
 3491 class. The qualifier *EmbeddedInstance* has been introduced because of this, and it should be  
 3492 used in the CIM Schema instead of *EmbeddedObject*, where applicable.

- 3493 • There is no mapping for CIM Triggers.

3494 Reason: CIM Triggers, while defined in the CIM metamodel, are not used in the CIM Schema,  
 3495 nor is it clear how they would be used.

## 3496 5.21 Definition of UML Profiles and Type Libraries

3497 Beginning with UML 2.0, UML profiles are represented by the UML *Profile* metaclass, so they have  
3498 become a runtime entity that can be modified dynamically as new qualifiers are added.

3499 In this document, the term "UML profile" is used for the instances of the UML *Profile* metaclass, and for  
3500 the files that are the physical representation of the UML Profile in some format. If a specific notion of  
3501 "UML profile" is referred to, appropriate qualifications are used.

3502 Note that a UML type library is a normal UML user model that happens to contain type definitions and  
3503 therefore is called "type library".

3504 This subclause normatively defines the existence and contents of the UML profiles and UML type libraries  
3505 that represent the UML Profile for CIM. It references UML constructs defined in other subclauses when  
3506 defining the contents.

### 3507 5.21.1 Profile "CIM"

3508 There shall be a UML profile named "CIM".

3509 A UML user model that intends to become a CIM model shall apply the UML profile "CIM".

3510 It is left to the UML user model whether the UML profile "CIM" is applied strictly or non-strictly.

3511 The UML profile "CIM" shall define the following elements:

- 3512 • All meta element stereotypes, as defined in 5.15.
- 3513 • All qualifier scope stereotypes, as defined in 5.13.2.
- 3514 • All qualifier type stereotypes including any enumerations they use, as defined in 5.13.2. Note  
3515 that these stereotypes depend on the qualifier type declarations in CIM, so this causes the UML  
3516 profile "CIM" to be modified during the act of importing CIM qualifier type declarations.
- 3517 • All marker stereotypes, as defined in the subclauses of 5.13.4.
- 3518 • Any relations between these elements, as defined in this document.

3519 The UML profile "CIM" shall import the UML type library "CIMDatatypes".

### 3520 5.21.2 Profile "CIMQualifierType"

3521 There shall be a UML profile named "CIMQualifierType". This UML profile shall not be applied to any UML  
3522 user models. It shall be applied non-strictly to the UML profile "CIM".

3523 The UML profile "CIMQualifierType" shall define the following elements:

- 3524 • The stereotype *CIM\_QualifierType*, as defined in Table 13.
- 3525 • The stereotype *CIM\_QualifierInheritanceRule*, as defined in Table 14.
- 3526 • Any relations between these elements, as defined in this document.

### 3527 5.21.3 Type Library "CIMDatatypes"

3528 There shall be a UML type library named "CIMDatatypes". It shall have the stereotype "modelLibrary"  
3529 applied.

3530 A UML user model that intends to become a CIM model shall import the UML type library  
3531 "CIMDatatypes".

3532 The UML type library "CIMDatatypes" shall define the following elements:

- 3533       • The *DataType* metaclass instances defined in Table 8.
- 3534



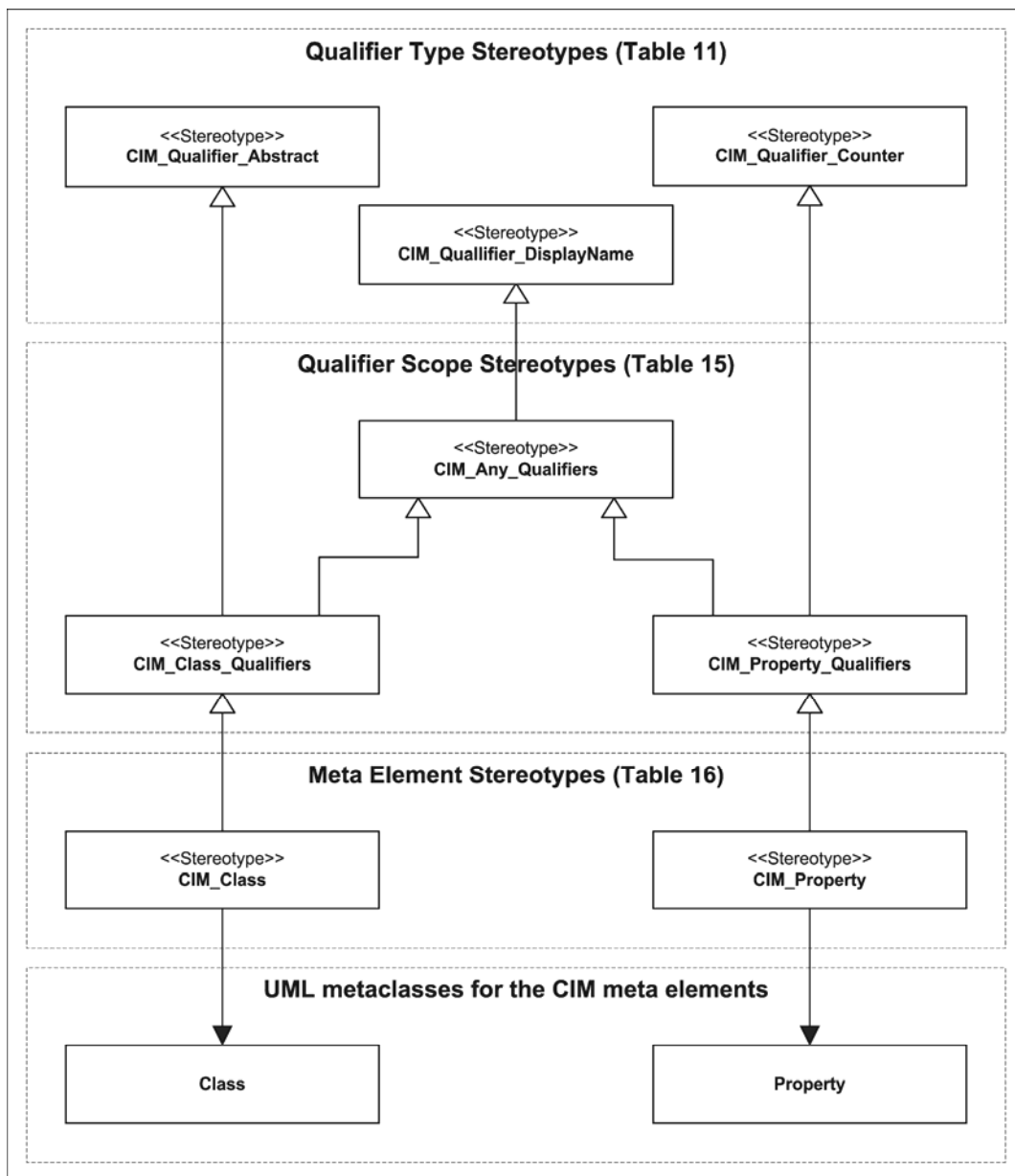
# ANNEX A (informative)

## UML Metamodel Diagrams

3535  
3536  
3537  
3538

3539 The diagrams in this appendix show the UML metaclasses involved in this document.

3540 Figure 3 shows the stereotype inheritance hierarchy resulting from the mapping of CIM qualifiers. The  
3541 stereotypes shown as an example are only those for CIM classes and CIM properties, and only a subset  
3542 of the qualifier types.

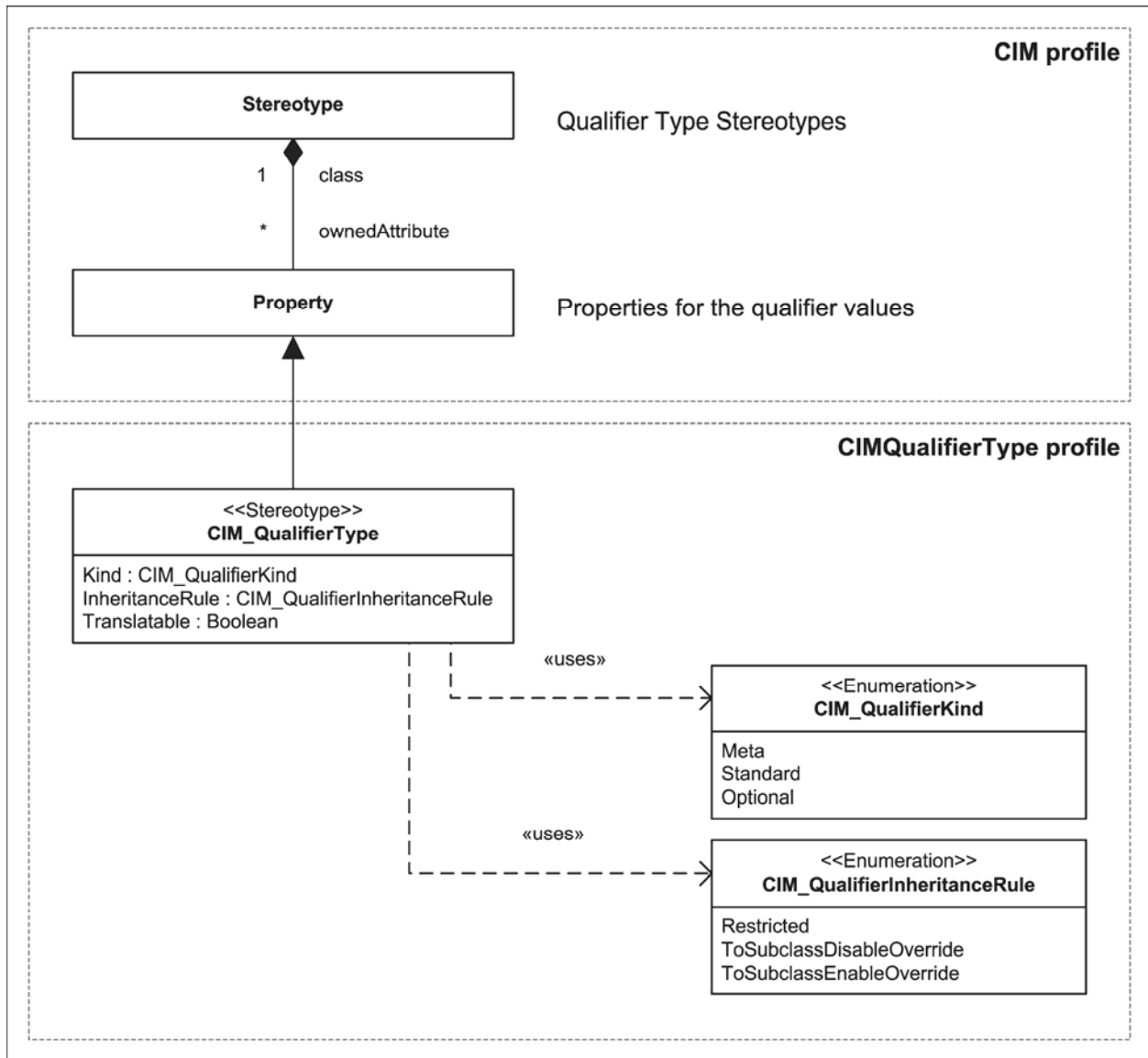


3543

3544

Figure 3 – Inheritance of Qualifier Related Stereotypes

3545 Figure 4 shows the contents of the UML profile "CIMQualifierType" as defined in 5.21.2.

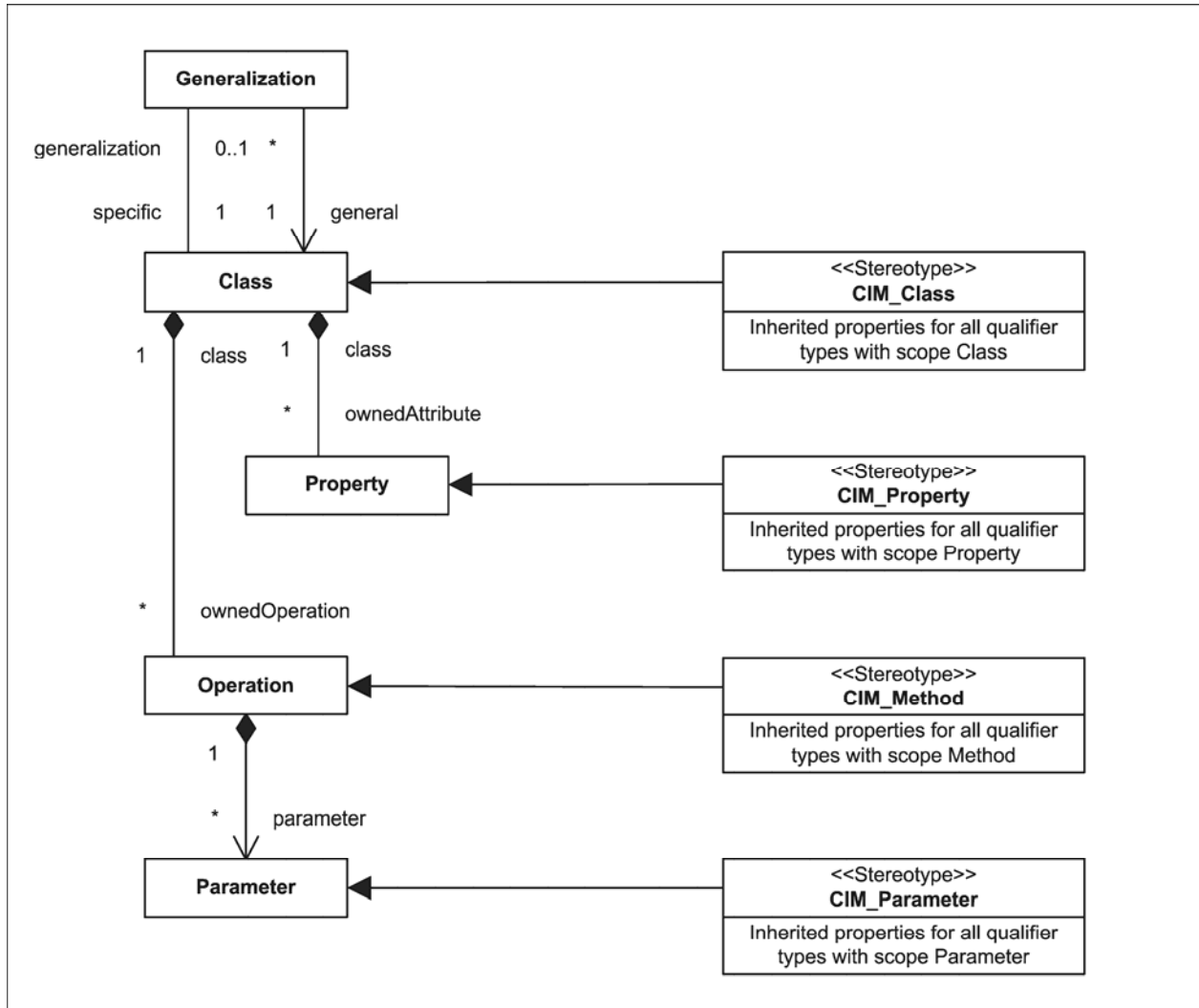


3546

3547

Figure 4 – Contents of UML Profile "CIMQualifierType"

3548 Figure 5 shows a high level view of the UML metaclasses involved in representing a normal (non-  
 3549 association, non-indication) CIM class. It does not show the UML metaclass instances for lower level  
 3550 elements such as DataTypes, Comments or Constraints. Between the elements it includes, it shows all  
 3551 UML meta-associations that are compositions but leaves away some non-compositions. Note that the  
 3552 elements labeled with "<<Stereotype>>" are stereotype definitions.

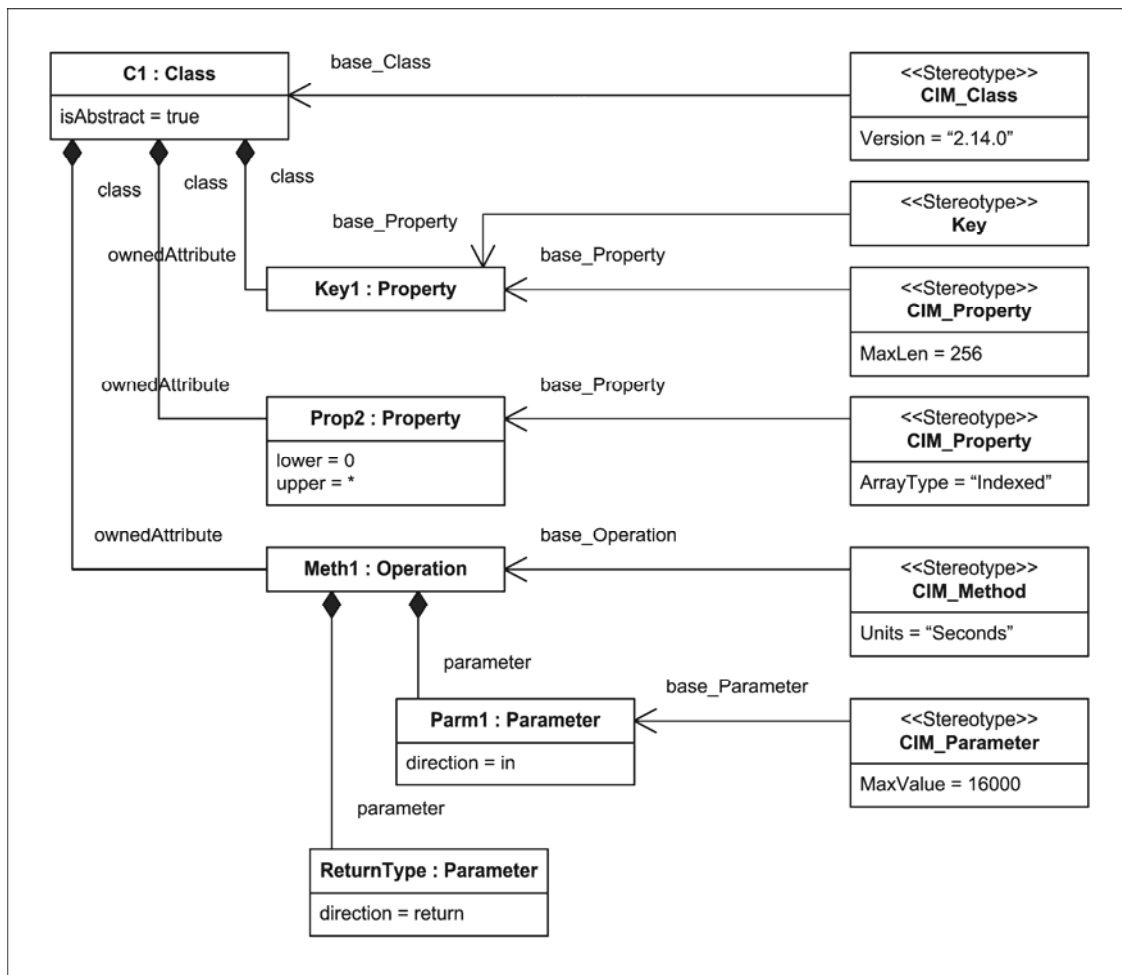


3553

3554

Figure 5 – UML Metaclasses Representing a CIM Class (high level)

3555 Figure 6 shows the UML metaclass instances representing a simple CIM class specified in the CIM MOF  
 3556 fragment shown in Figure 7. Again, some lower level elements have been left out for clarity. Note that the  
 3557 elements labeled with "<<Stereotype>>" are applied stereotypes, i.e., they are not stereotype definitions.



3558

3559

Figure 6 – UML Metaclass Instances for a Simple CIM Class

```
[Abstract, Version ("2.14.0")]
class XMP_C1 {

    [Key, MaxLen(256)]
    string Key1;

    [ArrayType("Indexed")]
    string Prop2 [];
}
```

3560

3561

Figure 7 – CIM MOF Definition of a Simple CIM Class

**ANNEX B  
(informative)**

**Change Log**

3562  
3563  
3564  
3565

3566

Version	Date	Author	Description
1.0.0	08/11/2009		DMTF Standard Release

## Bibliography

3567

3568 This section contains references to useful documents which are not normatively used by this document.

3569 *OMG MOF 2.0 XMI Mapping Specification, Version 2.1,*

3570 <http://www.omg.org/cgi-bin/doc?formal/05-09-01>

3571 *OMG MOF Core Specification, Version 2.0,*

3572 <http://www.omg.org/cgi-bin/doc?formal-06-01-01>

3573 *OMG MOF 2.0 Query / Views / Transformations Specification,*

3574 <http://www.omg.org/cgi-bin/doc?ptc/2005-11-01>

3575