



1

2

3

4

Document Number: DSP0230

Date: 2011-06-30

Version: 1.1.0

5

WS-CIM Mapping Specification

6

Document Type: Specification

7

Document Status: DMTF Standard

8

Document Language: en-US

9

10 Copyright notice

11 Copyright © 2007, 2011 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

32

CONTENTS

34	Foreword	5
35	Introduction	6
36	1 Scope	7
37	1.1 In-Scope Features	7
38	1.2 Out-of-Scope Considerations	7
39	2 Conformance	8
40	3 Normative References	8
41	4 Terms and Definitions	9
42	5 Symbols and Abbreviated Terms	11
43	6 Namespace Prefixes and Schema Locations	11
44	7 Dereferencing Schema URI Locations in Order to Access XML Schema	13
45	8 Mapping Primitive Datatypes	14
46	8.1 Escaping String Values in XML	15
47	8.2 cimDateTime Datatype	15
48	8.3 CIM References	16
49	8.4 cimAnySimpleType Datatype	17
50	8.5 Derivation by Restriction	17
51	8.6 WS-CIM Canonical Values	18
52	9 CIM Class to XML Schema Mappings	18
53	9.1 Class Namespace	18
54	9.2 Properties	19
55	9.3 Class Structure	27
56	9.4 Class Inheritance	30
57	9.5 Method Parameters	30
58	9.6 CIM Instances	33
59	9.7 Superclass Class Representation	34
60	10 CIM Methods to WSDL Mappings	35
61	10.1 Defining WSDL Message Structures	35
62	10.2 Defining WSDL Operation Structures	36
63	10.3 Defining wsa:Actions	36
64	11 Qualifier Mappings	37
65	11.1 General Format	37
66	11.2 Mapping CIM Qualifiers to XSD Elements	40
67	11.3 Inheritance of Qualifiers	42
68	12 Qualifier Annotations	42
69	12.1 Extension Qualifiers	44
70	12.2 Qualifier Values	44
71	12.3 Additional Information Not Directly From Qualifiers	47
72	ANNEX A (Informative) Schemas	49
73	A.1 Common WS-CIM Schema: DSP8004	49
74	A.2 Qualifiers Schema: DSP8005	52
75	A.3 Class Hierarchy Type Schema: DSP8006	54
76	ANNEX B (Informative) Examples	55
77	B.1 MOF Definitions	55
78	B.2 XSD	56
79	B.3 WSDL Fragments	61
80	B.4 MetaData Fragments	61
81	ANNEX C (Informative) Collation Optimization Available to Implementers	65
82	C.1 Sorting with Limited Character Set	65
83	C.2 Note of Caution Concerning Collation	65

84	ANNEX D (Informative) Example Schema With Qualifier Annotations.....	66
85	ANNEX E (Informative) Change Log	101
86	Bibliography	102
87		

88 Tables

89	Table 1 – Namespaces	11
90	Table 2 – Namespace Prefixes	12
91	Table 3 – Schema URI Locations	12
92	Table 4 – XSD DSP Numbers.....	12
93	Table 5 – Mapping CIM Datatypes to WS-CIM Datatypes	14
94	Table 6 – Rules for Converting datetime to cimDateTime	15
95	Table 7 – Rules for Converting cimDateTime to datetime	16
96	Table 8 – CIM Qualifiers Mapped to XSD Elements.....	40
97	Table 9 – Rules of Qualifier Inheritance.....	42
98		

Foreword

100 The *WS-CIM Mapping Specification* (DSP0230) was prepared by the DMTF WS-Management Working
101 Group.

102 The authors would like to acknowledge Andrea Westerinen (employed by Cisco at the time and now at
103 Microsoft) for drafting the Charter of the Working Group and initially leading the effort as Chairperson.

104 Authors:

- 105 • Akhil Arora, Sun Microsystems, Inc.
- 106 • Ed Boden, IBM
- 107 • Mark Carlson, Sun Microsystems, Inc. (past Co-Chair)
- 108 • Josh Cohen, Microsoft Corporation
- 109 • Jim Davis, WBEM Solutions, Inc.
- 110 • Asad Faizi, Microsoft Corporation (past Editor)
- 111 • Steve Hand, Symantec Corporation (past Editor and Chair)
- 112 • Vincent Kowalski, BMC Software, Inc. (past Co-Chair)
- 113 • Heather Kreger, IBM
- 114 • Richard Landau, Dell Inc. (Editor)
- 115 • Tom Maguire, EMC
- 116 • Andreas Maier, IBM
- 117 • James Martin, Intel Corporation
- 118 • Bryan Murray, Hewlett-Packard
- 119 • Brian Reistad, Microsoft Corporation
- 120 • Mitsunori Satomi, Hitachi
- 121 • Hemal Shah, Broadcom
- 122 • Sharon Smith, Intel Corporation
- 123 • Kirk Wilson, CA, Inc. (past Editor)
- 124 • Dr. Jerry Xie, Intel Corporation

125

126

Introduction

127 Management based on the Common Information Model (CIM) in a Web Services environment requires
128 that the CIM Schema (classes, properties, and methods) be rendered in XML Schema and Web Services
129 Description Language (WSDL). To achieve this, CIM must be mapped to WSDL and XML Schema
130 through an explicit algorithm that can be programmed for automatic translation.

131 This specification provides the normative rules and recommendations that describe the structure of the
132 XML Schema, WSDL fragments, and metadata fragments that correspond to the elements of CIM
133 models, and the representation of CIM instances as XML instance documents. A conformant
134 implementation of a CIM model to XML Schema, WSDL fragments, and metadata fragments
135 transformation algorithm must yield an XML Schema, WSDL fragments, and metadata fragments as
136 described in this specification. These CIM models may be expressed in CIM Managed Object Format
137 (MOF) or in other equivalent ways. Throughout this specification, examples illustrate the mapping from
138 CIM MOF.

139 Document Conventions

140 In XML and MOF examples, an ellipsis (" . . . ") indicates omitted or optional entries that would typically
141 occupy the position of the ellipsis.

142 The following conventions are followed for defining formats of entries such as URIs:

- 143 • Literal characters within a format definition are surrounded by single quotes.
- 144 • Names of variables within a format are in standard text and are explicitly defined by means
145 of a "Where: variable-name is ..." section that follows the format definition.
- 146 • A specific value of a variable within a generalized example of a formatted entry is displayed
147 in *italics*.
- 148 • Definitions of formats are case sensitive.
- 149 • Whitespace, if any, in formats is explicitly indicated.

150 The following typographical conventions are used:

- 151 • `Monospace font`: CIM datatypes and element names as well as XML and WSDL
152 element and attribute names.
- 153 • `Courier new 8, gray background`: Code examples

154

155

WS-CIM Mapping Specification

1 Scope

157 The goal of this specification is to produce a normative description of a protocol-independent mapping of
158 CIM models to XML Schema, WSDL fragments, and metadata fragments. The features of CIM that are
159 within the scope of this specification correspond to a subset of the features of CIM that are defined in the
160 *CIM Infrastructure Specification*, [DSP0004](#).

161 Another goal of this specification is to allow the most expedient use of current Web Services (WS)
162 infrastructure as a foundation for implementing a WS-CIM compliant system. This specification has been
163 written to leverage the existing Web Services standards and best practices that are currently widely
164 deployed and supported by Web Services infrastructure. As those standards and best practices evolve,
165 future versions of this specification should evolve to include them.

1.1 In-Scope Features

167 The following XML Schema, WSDL, and metadata is defined for the Common Information Model (CIM):

- 168 • Namespace URIs and the XML Schema definitions for CIM classes and their properties,
169 qualifiers, and methods. The mapping of CIM classes covers regular, association,
170 exception, and indication classes.
- 171 • Annotation sections that can be added to XML schema definition files to describe the
172 qualifiers of CIM classes and their properties.
- 173 • WSDL message definitions for CIM methods. The WSDL mapping supports WSDL version
174 1.1.
- 175 • WSDL portType operation definitions for CIM methods
- 176 • Metadata fragments for CIM qualifiers to be accessed separately from the XML schema.
177 The representations of qualifier information in these metadata fragments and the schema
178 annotations are different.

1.2 Out-of-Scope Considerations

180 The following items are outside the scope of this specification:

- 181 • This specification does not address mapping XML Schema structures to other CIM
182 representations, such as MOF or CIM-XML.
- 183 • Features excluded from the scope of this mapping include mapping of CIM instance
184 definitions in MOF and MOF compiler directives (pragmata). (Note that the mapping of CIM
185 instances is addressed in 9.6.)
- 186 • A WSDL mapping with portTypes and bindings is not provided by this specification. WSDL
187 bindings are protocol specific.
- 188 • Protocol-specific features of CIM or Web-Based Enterprise Management (WBEM), such as
189 CIM Operations over HTTP, the XML Representation of CIM, or WS-Management, are
190 outside the scope of this specification.
- 191 • This version of the specification does not provide mappings for Qualifier declarations. This
192 version is limited to the XML Schema definitions of metadata instances (Qualifier values)
193 that correspond to the CIM qualifiers in a CIM model.

- 194 • This version does not specify a metadata container for the mapped Qualifier values, but
195 leaves it to the specific protocol to determine where metadata resides.
- 196 • This version of the specification does not allow distinguishing empty arrays from arrays that
197 are NULL. This limitation is a result of the decision to use existing standards, which use
198 inline arrays, for representing arrays in XML.
- 199 • The invocation of CIM methods may result in an exception represented by one or more
200 instances of classes whose `EXCEPTION` qualifiers are effectively TRUE. While such
201 instances shall be represented in XML according to the mapping rules for CIM classes in
202 clause 9, requirements regarding the transmittal of these exceptions when they occur are
203 protocol-specific and are not in scope for this specification.

204 **2 Conformance**

205 To be compliant with this specification, an XML Schema, WSDL fragment definitions (messages and
206 operations), and metadata elements shall conform to all normative requirements of this specification.

207 Implementations shall not use the namespaces for CIM classes that conform to this specification (see 9.1)
208 unless the XML Schema for those classes conforms to this specification.

209 **3 Normative References**

210 The following referenced documents are indispensable for the application of this document. For dated or
211 versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies.
212 For references without a date or version, the latest published edition of the referenced document
213 (including any corrigenda or DMTF update versions) applies.

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

216 DMTF DSP0226, *Web Services for Management Specification 1.1*,
217 http://www.dmtf.org/standards/published_documents/DSP0226_1.1.pdf

218 DMTF DSP8004, *WS-CIM Common XSD 1.0*,
219 <http://schemas.dmtf.org/wbem/wscim/1/common.xsd>

220 DMTF DSP8005, *WS-CIM Qualifiers XSD 1.0*,
221 <http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/qualifiers.xsd>

222 DMTF DSP8006, *WS-CIM ClassHierType XSD 1.0*,
223 <http://schemas.dmtf.org/wbem/wscim/1/classhiertype.xsd>

224 IETF RFC 3987, *Internationalized Resource Identifiers (IRIs)*, January 2005,
225 <http://www.ietf.org/rfc/rfc3987.txt>

226 IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, January 2005,
227 <http://www.ietf.org/rfc/rfc3986.txt>

228 UNICODE COLLATION ALGORITHM, Unicode Technical Standard #10, version 5.1.0, 2008-03-28
229 <http://Unicode.org/reports/tr10>

230 ISO, ISO/IEC 10646:2003 *Information technology – Universal Multiple-Octet Coded Character Set (UCS)*,
231 [http://standards.iso.org/ittf/PubliclyAvailableStandards/c039921_ISO_IEC_10646_2003\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c039921_ISO_IEC_10646_2003(E).zip)

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

234 W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation 26 November 2008,
235 <http://www.w3.org/TR/xml/>

236 W3C, *Namespaces in XML 1.0 (Third Edition)*, W3C Recommendation, 8 December 2009,
237 <http://www.w3.org/TR/REC-xml-names>

238 W3C, *Web Services Addressing (WS-Addressing) 1.0 – Core*, W3C Recommendation, 9 May 2006,
239 <http://www.w3.org/TR/ws-addr-core/>

240 W3C, *Web Services Description Language (WSDL) 1.1*, W3C Note, 15 March 2001,
241 <http://www.w3.org/TR/wsdl>

242 W3C, *XML Schema Part 2: Datatypes*, W3C Recommendation, October 2004,
243 <http://www.w3.org/TR/xmlschema-2/>

244 W3C, *XML Schema Part 1: Structures*, W3C Recommendation, October 2004,
245 <http://www.w3.org/TR/xmlschema-1/>

246 **4 Terms and Definitions**

247 For the purposes of this document, the following terms and definitions apply.

248 **4.1**

249 **can**

250 used for statements of possibility and capability, whether material, physical, or causal

251 **4.2**

252 **cannot**

253 used for statements of possibility and capability, whether material, physical or causal

254 **4.3**

255 **conditional**

256 indicates requirements to be followed strictly in order to conform to the document when the specified
257 conditions are met

258 **4.4**

259 **mandatory**

260 indicates requirements to be followed strictly in order to conform to the document and from which no
261 deviation is permitted

262 **4.5**

263 **may**

264 indicates a course of action permissible within the limits of the document

265 **4.6**

266 **need not**

267 indicates a course of action permissible within the limits of the document

268 **4.7**

269 **optional**

270 indicates a course of action permissible within the limits of the document

- 271 **4.8**
272 **shall**
273 indicates requirements to be followed strictly in order to conform to the document and from which no
274 deviation is permitted
- 275 **4.9**
276 **shall not**
277 indicates requirements to be followed strictly in order to conform to the document and from which no
278 deviation is permitted
- 279 **4.10**
280 **should**
281 indicates that among several possibilities, one is recommended as particularly suitable, without
282 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 283 **4.11**
284 **should not**
285 indicates that a certain possibility or course of action is deprecated but not prohibited
- 286 **4.12**
287 **Global Element Declaration**
288 element declaration in an XML Schema that places the element as an immediate child of the root element
289 of the schema
- 290 **4.13**
291 **Managed Object Format**
292 an IDL based language, defined by the DMTF, expressing the structure, behavior, and semantics of a
293 CIM class and its instances
- 294 **4.14**
295 **Uniform Record Identifier**
296 a Uniform Resource Identifier (URI) is a compact sequence of characters that identifies an abstract or
297 physical resource. See [RFC3986](#).
- 298 **4.15**
299 **Runtime**
300 describes the situation where XML instances are produced that are conformant to this specification
- 301 **4.16**
302 **WS-CIM**
303 of or pertaining to this specification
304 NOTE: "WS-CIM" is not an acronym; it should be treated simply as the name of the contents of the specification.
- 305 **4.17**
306 **XML instance document**
307 an XML document that conforms to a specified XML Schema
308 As used in this specification, *XML instance document* refers to a document that conforms to an XML
309 Schema that conforms to the rules in clause 9.
- 310 **4.18**
311 **XSDL**
312 offers facilities for describing the structure and constraining the contents of XML documents, including
313 those which exploit the XML Namespace facility. XSDL documents have the '.xsd' file extension. See
314 [XML Schema Part 1: Structures](#).

315 **5 Symbols and Abbreviated Terms**

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

317 **5.1**

318 **GED**

319 Global Element Declaration

320 **5.2**

321 **MOF**

322 Managed Object Format

323 **5.3**

324 **URI**

325 Uniform Resource Identifier

326 **5.4**

327 **WSDL**

328 Web Services Description Language

329 **5.5**

330 **XML**

331 Extensible Mark-up Language

332 **5.6**

333 **XSD**

334 XML Schema Definition

335 **6 Namespace Prefixes and Schema Locations**

336 Table 1 through Table 4 list URIs using the ws-cim-major-version, *X*, and the cim-schema-major-version,
337 *Y*. When using these URIs, replace the *X* and *Y* variables with the actual version numbers.

338 This specification defines namespaces as shown in Table 1.

339 **Table 1 – Namespaces**

Namespace	Description
http://schemas.dmtf.org/wbem/wscim/ <i>X</i> /cim-schema/ <i>Y</i> / <i>ClassName</i>	Contains the schema for the class <i>ClassName</i>
http://schemas.dmtf.org/wbem/wscim/ <i>X</i> /common	Contains the schema for common elements such as datatypes required for defining XML schemas for CIM classes
http://schemas.dmtf.org/wbem/wscim/ <i>X</i> /cim-schema/ <i>Y</i> /qualifiers	Contains the schemas of qualifiers that are mapped to metadata fragments
http://schemas.dmtf.org/wbem/wscim/ <i>X</i> /classhiertype	Contains the schema definitions for representing the subclass/superclass hierarchy of the CIM Schema as the value of a property
http://schemas.dmtf.org/wbem/wscim/ <i>X</i> /cim-schema/ <i>Y</i> /classhierarchy	Contains the GEDs that represent the subclass/superclass hierarchy of the CIM Schema

340 The namespace prefixes shown in Table 2 are used throughout this document. Note that the choice of
 341 any namespace prefix is arbitrary and not semantically significant (see [NameSpaces in XML](#)).

342 **Table 2 – Namespace Prefixes**

Prefix	Namespace
class	http://schemas.dmtf.org/wbem/wscim/X/cim-schema/Y/ClassName
cim	http://schemas.dmtf.org/wbem/wscim/X/common
cimQ	http://schemas.dmtf.org/wbem/wscim/X/cim-schema/Y/qualifiers
ctype	http://schemas.dmtf.org/wbem/wscim/X/classhiertype
chier	http://schemas.dmtf.org/wbem/wscim/X/cim-schema/Y/classhierarchy
wsa	Any wsa:addressing standard defining EPRs, such as http://www.w3.org/2005/08/addressing (see Web Services Addressing (WS-Addressing) 1.0 – Core) or http://schemas.xmlsoap.org/ws/2004/08/addressing (see DSP0226 , "Management Addressing" clause)
wSDL	http://schemas.xmlsoap.org/wSDL (see Web Services Description Language (WSDL) 1.1)
xs	http://www.w3.org/2001/XMLSchema (see XML Schema Parts 1 & 2)
xsi	http://www.w3.org/2001/XMLSchema-instance

343 Table 3 defines the schema location URIs for the schemas defined in this specification.

344 **Table 3 – Schema URI Locations**

Prefix	Schema Location URLs
class	http://schemas.dmtf.org/wbem/wscim/X/cim-schema/Y/ClassName.xsd
cim	http://schemas.dmtf.org/wbem/wscim/X/common.xsd
cimQ	http://schemas.dmtf.org/wbem/wscim/X/cim-schema/Y/qualifiers.xsd
ctype	http://schemas.dmtf.org/wbem/wscim/X/classhiertype.xsd
chier	http://schemas.dmtf.org/wbem/wscim/X/cim-schema/Y/classhierarchy.xsd

345 Table 4 defines the DSP numbers for the XSD files defined by this specification.

346 **Table 4 – XSD DSP Numbers**

XSD File Name	DSP Number
http://schemas.dmtf.org/wbem/wscim/X/common.xsd	DSP8004
http://schemas.dmtf.org/wbem/wscim/X/cim-schema/Y/qualifiers.xsd	DSP8005
http://schemas.dmtf.org/wbem/wscim/X/classhiertype.xsd	DSP8006

347 7 Dereferencing Schema URI Locations in Order to Access XML 348 Schema

349 This clause defines how DMTF is to publish artifacts produced in accordance with this specification.

350 A client application may construct schema URIs as specified in the following subclause to retrieve the
351 schema documents from the DMTF schema website (<http://schemas.dmtf.org/>).

352 DMTF shall publish the XML schema documents listed in Table 3 at the URI locations specified in Table
353 3. A schema document published at one of these URI locations will always represent the most recent
354 version of the class namespace definition.

355 DMTF shall also publish productions of CIM classes in XSD schema at locations that support retrieval of
356 specific versions of the class namespace definition as follows:

- 357 • At a URI location where the ws-cim-major-version number in the URI specified in Table 3 is
358 replaced with the major, minor, and revision formatted as “major [“.” Minor [“.” Revision
359]]” of the exact CIM schema version of which the class is a member. All classes published
360 at this URI location shall be final classes.

361 EXAMPLE:

362 <http://schemas.dmtf.org/wbem/wscim/1.1.0/cim-schema/2.17.0/ClassName.xsd>

- 363 • At a URI location where the ws-cim-major-version number in the URI specified in Table 3 is
364 replaced with the major, minor, and revision numbers of the CIM schema version and
365 where a “plus” character (+) is appended to that version number. The format shall be: “
366 major [“.” Minor [“.” Revision]] ”+. Each such class shall include all experimental content
367 defined for the included version of that class.

368 EXAMPLE:

369 <http://schemas.dmtf.org/wbem/wscim/1.1.0/cim-schema/2.17.0+/ClassName.xsd>

370 EXAMPLE 1: If the latest available final version of the CIM schema is 2.11.0 and the WS-CIM
371 mapping specifications is 1.3.0, the following URI locations would retrieve the same XML Schema
372 file:

373 <http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/ClassName.xsd>

374 <http://schemas.dmtf.org/wbem/wscim/1.3.0/cim-schema/2.11.0/ClassName.xsd>

375 EXAMPLE 2: To retrieve the XML Schema for the same class from CIM schema version 2.10.1
376 based on WS-CIM mapping version 1.2.0, the following URI location would be used:

377 <http://schemas.dmtf.org/wbem/wscim/1.2.0/cim-schema/2.10.1/ClassName.xsd>

378 EXAMPLE 3: To retrieve the XML Schema for the same class from CIM schema version 2.11.0
379 “experimental” based on WS-CIM mapping version 1.3.0, the following URI location could be used:

380 <http://schemas.dmtf.org/wbem/wscim/1.2.0/cim-schema/2.10.1+/ClassName.xsd>

381 7.1.1 Validation of CIM Instances

382 In some cases, it is desirable to attempt schema validation of the Instance document. However, if the
383 major version of the class or other XML Schemas defined in this specification that are used in the
384 instance document differ from the version of the XML Schemas that the recipient of the instance
385 document has, then validation is impossible. If the major version is the same and the minor version
386 differs, validation is possible.

387 DMTF permits the structure of the class to change in backwards-compatible ways within a release of a
388 major version of CIM. [DSP0004](#) (see “Schema Versions”) describes the nature of permitted changes in

389 this case. However, the changes permitted could cause XML schema validation errors. Implementations
 390 have a choice on how to be resilient to the changes permitted in such cases.

391 To perform conventional XML schema validation, a validator must obtain the exact schema version used
 392 to produce the instance. If the exact class schema document for the received CIM instance is known, it
 393 may be retrieved as described previously. The Instance document may indicate the URI location for the
 394 Class schema document to which its structure conforms through the xsi:schemaLocation attribute as
 395 specified in 9.6. Validation of the CIM instance document with the class schema document retrieved
 396 through this mechanism shall be possible.

397 Alternatively, the recipient may use a custom XML schema validation routine that tolerates the permitted
 398 backwards-compatible changes previously referenced.

399 8 Mapping Primitive Datatypes

400 Specific WS-CIM datatypes are defined as extensions of simple XSD datatypes. These extended
 401 WS-CIM datatypes allow the use of any attribute in conjunction with the simple XSD base datatype that
 402 corresponds directly to a CIM datatype. The WS-CIM datatypes are defined in the common.xsd file
 403 (ANNEX A).

404 CIM datatypes are converted to WS-CIM datatypes as shown in Table 5.

405 **Table 5 – Mapping CIM Datatypes to WS-CIM Datatypes**

CIM Datatype	Corresponding Base XSD Datatypes	WS-CIM Datatypes
uint8	xs:unsignedByte	cim:cimUnsignedByte
sint8	xs:byte	cim:cimByte
uint16	xs:unsignedShort	cim:cimUnsignedShort
sint16	xs:short	cim:cimShort
uint32	xs:unsignedInt	cim:cimUnsignedInt
sint32	xs:int	cim:cimInt
uint64	xs:unsignedLong	cim:cimUnsignedLong
sint64	xs:long	cim:cimLong
string	xs:string	cim:cimString
boolean	xs:boolean	cim:cimBoolean
real32	xs:float	cim:cimFloat
real64	xs:double	cim:cimDouble
datetime	xs:duration xs:date xs:time xs:dateTime xs:string Depending on use-case See 8.2.	cim:cimDateTime
char16	xs:string With maxLength restriction = 1	cim:cimChar16
<class> REF	N/A	cim:cimReference See 8.3.

406 NOTE: For mapping of array properties, see 9.2.2.

407 CIM properties that are designated with the following qualifiers require special mapping that supersedes
 408 the mappings shown in Table 5:

- 409 Octetstring
- 410 EmbeddedInstance
- 411 EmbeddedObject

412 See 9.2.4 for mapping of Octetstring properties; see 9.2.5 for mapping of EmbeddedInstance and
 413 EmbeddedObject properties.

414

415 8.1 Escaping String Values in XML

416 String values can contain characters and sequences that must or should be avoided or escaped to be
 417 used safely in XML. XML specifies the necessary treatments of such characters and sequences: some
 418 characters must be escaped, and others may be, as described in the W3C XML Specification section 2.4.

- 419 • The following characters in string values should be escaped using numeric character references:
 420 linefeed (0x0A), carriage return (0x0D), and horizontal tab (0x09). In particular, escaping
 421 carriage return (0x0D) is strongly recommended.

422 Additionally, string values may be represented by a service using CDATA sections (i.e., <![CDATA[. . .
 423]]>) to encapsulate text that may contain characters that must be avoided in XML, as described in the
 424 W3C XML Specification section 2.7. A client consumer of an XML schema or instance must support XML
 425 un-escaping with numeric character references and CDATA.

426

427 8.2 cimDateTime Datatype

428 The `cim:cimDateTime` datatype is defined as follows:

```

429 <xs:complexType name="cimDateTime">
430   <xs:choice>
431     <xs:element name="CIM_DateTime" type="xs:string" nillable="true"/>
432     <xs:element name="Interval" type="xs:duration"/>
433     <xs:element name="Date" type="xs:date"/>
434     <xs:element name="Time" type="xs:time"/>
435     <xs:element name="Datetime" type="xs:dateTime"/>
436   </xs:choice>
437 <xs:anyAttribute namespace="##any" processContents="lax"/>
438 </xs:complexType>
    
```

439 The rules shown in Table 6 should be used to convert CIM `datetime` to `cim:cimDateTime`.

440 **Table 6 – Rules for Converting datetime to cimDateTime**

CIM datetime Use Case	String Condition	cim:cimDateTime Element
interval	String contains ":"	Interval
date and time	String contains "+" or "-" and does not contain any asterisks	Datetime
time	String contains "+" or "-" and no asterisks in the hhmmss.mmmmmm portion, and only asterisks in the yyyyymmdd portion	Time
date	String contains "+" or "-" and no asterisks in the yyyyymmdd portion, and only asterisks in the hhmmss.mmmmmm portion	Date
Other	String asterisks other than indicated above	CIM_DateTime

441 The rules shown in Table 7 should be used to convert `cimDateTime` elements on the client side to their
 442 representation in CIM.

443 **Table 7 – Rules for Converting `cimDateTime` to `datetime`**

<code>cim:cimDateTime</code> Element	Representation of <code>datetime</code> in CIM
Interval	CIM <code>datetime</code> that is an Interval. Fields that are not significant shall be replaced with asterisks. For example, an interval of 2 days 23 hours would be converted to 0000000223****.*****:000
Datetime	CIM <code>datetime</code> that is a time stamp. The mapping of timezone offset is left to the implementer to either normalize it to zero or preserve it in translation. Note that the resulting CIM <code>datetime</code> string does not contain any asterisks, because this XML element is used only when the original CIM <code>datetime</code> satisfies this condition.
Time	CIM <code>datetime</code> that is a time stamp. The mapping of timezone offset is left to the implementer to either normalize it to zero or preserve it in translation. Note that the resulting CIM <code>datetime</code> string does not contain any asterisks in the <code>hhmmss.mmmmmm</code> portion, and contains only asterisks in the <code>yyyymmdd</code> portion, because this XML element is used only when the original CIM <code>datetime</code> satisfies this condition.
Date	CIM <code>datetime</code> that is a time stamp. The mapping of timezone offset is left to the implementer to either normalize it to zero or preserve it in translation. Note that the resulting CIM <code>datetime</code> string does not contain any asterisks in the <code>yyyymmdd</code> portion, and contains only asterisks in the <code>hhmmss.mmmmmm</code> portion, because this XML element is used only when the original CIM <code>datetime</code> satisfies this condition.
<code>CIM_DateTime</code>	CIM <code>datetime</code> with a string equal to the XML element text.

444 8.3 CIM References

445 The `cim:cimReference` datatype is defined as follows:

```
446 <xs:complexType name="cimReference">
447   <xs:sequence>
448     <xs:any namespace="##other" maxOccurs="unbounded" processContents="lax"/>
449   </xs:sequence>
450   <xs:anyAttribute namespace="##any" processContents="lax"/>
451 </xs:complexType>
```

452 The `xs:any` element in this definition represents a structure of a single transport reference that uniquely
 453 identifies a location to which messages may be directed for the referenced entity. This structure may be
 454 either a single element that expresses the complete transport reference or a sequence of elements, if the
 455 transport reference requires multiple elements to uniquely identify a location. In the case of Addressing
 456 (see [Web Services Addressing \(WS-Addressing\) 1.0 – Core](#) and [DSP0226](#), "Management Addressing"
 457 clause), the `xs:any` element shall be replaced by the required `wsa:EndpointReference` child elements
 458 defined by Addressing recommendations, as if the property element were of type
 459 `wsa:EndpointReferenceType`. These requirements for the representation of the reference datatype
 460 supersede any requirements specified in [DSP0004](#) regarding the syntactical representation of a value of
 461 type reference.

462 The attribute `maxOccurs="unbounded"` shall not be misconstrued as allowing multiple transport
 463 references.

464 EXAMPLE: An example of the use of Addressing versions as a transport reference, mapped to the
 465 `AssociatedComponent` property, is as follows:

```
466 <xs:element name="AssociatedComponent" type="cim:cimReference"/>
```


467 The reference could appear in an XML instance document as in either of the following examples:

```
468 <AssociatedComponent
469     xmlns:wsa="http://www.w3.org/2005/08/addressing">
470     <wsa:Address>. . .</wsa:Address>
471     . . . <!-- Other EPR elements as defined in the 2005/08 specification -->
472 </AssociatedComponent>
```

```
474 <AssociatedComponent
475     xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
476     <wsa:Address>. . .</wsa:Address>
477     . . . <!-- Other EPR elements as defined in the 2004/08 specification -->
478 </AssociatedComponent>
```

479 8.4 cimAnySimpleType Datatype

480 WS-CIM also introduces a special type built on `xs:anySimpleType`, `cimAnySimpleType`.
 481 `cimAnySimpleType` extends `xs:anySimpleType` with the facility to add any attribute to an instance of
 482 this type in an XML instance document. This special datatype is required in the mapping of CIM
 483 properties with ValueMaps containing ranges because of a restriction in the XML Schema specification.
 484 CIM properties with ValueMaps containing ranges are mapped as restrictions of a WS-CIM datatype
 485 where the restriction contains an `xs:union` consisting of an explicit enumeration of any discrete values
 486 (if any) and the specific ranges specified by the ValueMap (see 9.2.3 for the mapping rules for properties
 487 with ValueMaps). However, XML Schema currently requires that the content of a restriction be the same
 488 as or be derived from the content type of the parent complex type that is being restricted. Because an
 489 XSD union can be of any XSD simple type, XML Schema restricts the use of a union in a derivation by
 490 restriction to a parent type whose content is of any simple type. Thus, CIM properties with ValueMaps
 491 containing ranges are mapped to XSD elements of the `cimAnySimpleType` datatype.

492 However, this mapping overrides the normal datatype mapping of the CIM property (as mapped in
 493 Table 5). Using the `cimAnySimpleType` datatype means that standard WS-CIM datatyping information
 494 is lost for the property. Consequently, the following normative rule governs the use of this special
 495 datatype:

496 The `cimAnySimpleType` datatype shall be used only for mapping CIM properties with ValueMaps
 497 containing ranges. Any other use of this datatype is considered non-conformant to the WS-CIM
 498 specification.

499 8.5 Derivation by Restriction

500 The purpose of the WS-CIM datatypes is to provide the ability to add any attributes to CIM data in the
 501 instance document where those attributes are not defined in the XSD definition of the data. For example:

```
502 . . .
503 <this:Name xmlns:AdditionalAttribute=". . .">
504     myName
505 </this:Name>
506 . . .
```

507 where `AdditionalAttribute` is a global attribute defined in a namespace (`xns`) and is not explicitly specified
 508 in the type definition of `Name`. Including the `AdditionalAttribute` attribute in the instance document is valid
 509 based on the presence of the following wildcard specification in the definition of the type for this data:

```
510 <xs:anyAttribute namespace="##any" processContents="lax"/>
```

511 However, the anyAttribute wildcard is not inherited by a type definition that is derived by restriction from a
512 parent type containing the wildcard. Therefore, the following normative rule applies to all derivations by
513 restriction:

514 To preserve attribute extensibility, the anyAttribute wildcard shall be specified in any derivation by
515 restriction from a WS-CIM datatype.

516 NOTE: All mapping rules involving a derivation by restriction in this specification explicitly stipulate the inclusion of
517 the anyAttribute wildcard in the mapping.

518 8.6 WS-CIM Canonical Values

519 The WS-CIM specification maps a CIM Boolean to an XSD Boolean. According to XSD data types
520 (<http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#boolean>), a Boolean value can be one of four
521 possible values: true, false, 1, or 0.

522 To promote interoperability, the WS-CIM specification requires adoption of canonical representation by
523 W3C XPath spec for XSD Boolean type. The implementations shall use the values of TRUE and FALSE
524 as the canonical values for Boolean type.

525 9 CIM Class to XML Schema Mappings

526 This clause contains the normative rules for mapping CIM elements into structures of XML Schema. Each
527 clause provides the following information:

- 528 • complete normative rules
- 529 • an example of the use of those rules
- 530 • runtime normative rules and examples, if necessary, to address runtime consideration

531 9.1 Class Namespace

532 Each CIM class has an assigned XML namespace, the *class namespace*. This clause defines the class
533 namespace, and subsequent clauses define how the class namespace is used in the mapping.

534 The rules for specifying the XSD namespace of a CIM class are as follows:

- 535 • Each CIM class shall be assigned its own namespace in the XML schema.

536 `'http://schemas.dmtf.org/wbem/wscim/' wscim-major-version '/cim-schema/' cim-schema-major-`
537 `version '/' cim-class-schema '_' cim-class-name`

538 Where:

539 wscim-major-version is the major version number of this specification. Note that this
540 version number changes only if there are incompatible changes in the specification.

541 cim-schema-major-version is the major version number of the CIM schema version to
542 which the class being converted belongs. Note that this version number changes only if
543 there are incompatible changes in the CIM schema.

544 cim-class-schema is the CIM schema name of the class (for example, "CIM"). Note that the
545 schema name may be vendor specific in the case of vendor extensions to CIM classes.

546 cim-class-name is the name of the CIM class.

- 547 • The process and rules for the publication of the schema documents that define class
548 namespaces are defined in [DSP4009](#).

549 EXAMPLE: The `CIM_ComputerSystem` class that belongs to version 2.11.0 of the CIM schema would
550 have the following namespace:

551 http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ComputerSystem

552 9.2 Properties

553 This clause describes the general principles for converting CIM properties to XSD. It also describes
554 specific principles for converting array properties, value maps, octetstrings, and embedded objects and
555 instances.

556 9.2.1 General Principles

557 This clause defines and illustrates the normative rules that apply to the mapping of all CIM properties to
558 XSD.

559 9.2.1.1 General Rules

560 The rules for mapping CIM properties to XML Schema are as follows:

- 561 • Every property of a CIM class shall be represented by a global element definition. Note that
562 this rule applies to properties locally defined on the class itself and properties inherited
563 from superclasses (see 9.4). The GED that corresponds to a CIM property shall exhibit the
564 following features:
 - 565 – The GED shall be defined in the namespace of the class in the XML Schema (see 9.1).
 - 566 – The name of the GED shall be the same as the name of the CIM property.
 - 567 – The type of the GED shall comply with the datatype conversion table defined in clause 8.
568 However, in some cases, depending on what qualifiers apply to the CIM property, it is
569 necessary to restrict the default type of the GED element. The complete specification of the
570 type of the GED shall comply with the normative rules for mapping qualifiers. (See 11.2 for
571 the normative rules for mapping specific qualifiers to XSD structures.)
 - 572 • The GED of properties that are not arrays shall be specified with `nillable="true"`, if
573 the CIM property has a `Key` or `Required` qualifier with an effective value of `false`. The
574 GED of properties that are not arrays shall be specified without the `nillable` attribute
575 (the default of `nillable` is `false`), if the CIM property has a `Key` or `Required` qualifier
576 with an effective value of `true`. (See 11.3 for rules regarding the inheritance of qualifiers.)

577 NOTE: These rules do not apply to array properties. All GEDs that represent array properties shall be
578 specified with `nillable="true"`.

- 579 • The CIM Schema may assign default initializer values directly to properties, as, for
580 example, in the MOF construct `uint16 EnabledState=3` (see [DSP0004](#)). Default
581 initializer values shall be mapped to a metadata fragment using `<cim:DefaultValue>`.
582 Default initializer values shall not be mapped to the `xs:default` attribute, which carries
583 different semantics than CIM Schema default values.

584 The metadata fragment shall contain the `xsi:type` attribute, which specifies the primitive
585 datatype of the default initializer value. The specified datatype shall be the same as the base
586 type of the WS-CIM defined datatype of the XSD element that represents the CIM property
587 (see clause 8). The base type of a WS-CIM defined datatype shall be determined from its
588 datatype definition in the common namespace. In the case of the `cimDateTime` datatype,
589 `xsi:type` shall specify the primitive datatype of the element that is used to express the value
590 of `cimDateTime`.

591 NOTE: This rule precludes specifying default initializer values for REF properties (`cimReference`
 592 elements in XSD mapping). However, specifying such default initializer values is also unsupported in CIM.
 593 Values for `cimReference` elements can be provided only at runtime.

594 The GEDs that represent CIM properties are referenced by child elements within the element to which the
 595 CIM class that owns the properties is mapped (see 9.3). Rules for specifying the `minOccurs` and
 596 `maxOccurs` attributes of the elements that reference the GEDs are provided in 9.3.1.

597 EXAMPLE: As an example of the preceding rules, consider the following MOF fragment that defines the
 598 (hypothetical) class `EX_BaseComponent`. The complete definition of this class is presented in B.1.1.

```
599 class EX_BaseComponent {
600     datetime InstallDate;
601     [ ...
602         Required, MaxLen ( 1024 ) ]
603     string Name;
604     string StatusDescriptions[];
605     string HealthStatus;
606 };
```

607 Four GEDs need to be generated to represent the four properties of this class. Based on the preceding
 608 rules, the first two properties are mapped as follows:

```
609 <xs:element name="InstallDate" type="cim:cimDateTime" nillable="true"/>
610 <xs:element name="Name">
611     <xs:complexType>
612         <xs:simpleContent>
613             <xs:restriction base="cim:cimString">
614                 <xs:maxLength value="1024"/>
615                 <xs:anyAttribute namespace="##any" processContents="lax"/>
616             </xs:restriction>
617         </xs:simpleContent>
618     </xs:complexType>
619 </xs:element>
```

620 The complete mapping of this class is presented in B.2.1. For an example of a default initializer value
 621 metadata fragment, see B.4.2.

622 9.2.1.2 Runtime Rules for Attribute Value Assignment

623 The occurrence of properties in an XML instance document may be subject to the following rule:

- 624 • If a `Key` qualifier that has an effective value of `true` is associated with the CIM property,
 625 the `cim:Key` attribute may be applied to the corresponding element in the XML instance
 626 document. Use of the `cim:Key` attribute shall conform to the following rules:
 - 627 – If the attribute is present, its value shall be assigned as `true` in the XML instance.
 - 628 – If the application decides to apply the `cim:Key` attribute to the property, it shall apply it to
 629 all properties in the class that have a `Key` qualifier with an effective value of `true`
 630 associated with them. If a `Key` qualifier is not associated with the CIM property, this
 631 attribute shall be omitted.

632 The `Name` property can be designated with a `Key` qualifier in CIM:

```
633 class EX_SomeClass {
634     ...
635     [... Key]
636     string Name;
637 }
```

638 The instance document may specify the `cim:Key` attribute for this property as follows:

```
639 <EX_SomeClass>
640   ...
641   <Name cim:Key="true">MyName</Name>
642   ...
643 </EX_SomeClass>
```

644 The following clauses discuss the mapping of more complex CIM properties.

645 9.2.2 Array Properties

646 This clause defines and illustrates the specific rules for mapping CIM properties that are arrays.

647 9.2.2.1 General Rules

648 The rule for representing arrays is as follows:

649 Mapping of array properties shall follow the general principles for mapping properties in 9.2.1.1.

650 NOTE 1: Array properties have a multiplicity (`minOccurs` and `maxOccurs`) that corresponds to the specification of
651 their size in CIM. Rules for specifying the `minOccurs` and `maxOccurs` attributes to the element that represents an
652 array property are provided in 9.3.1.

653 NOTE 2: Inline arrays represent the current best practices and standards for mapping arrays to XML. New work is
654 beginning to explore alternative array-mapping strategies, and the committee shall track the progress of those efforts
655 for possible inclusion in future versions of this specification.

656 EXAMPLE 1: Consider the array `StatusDescriptions` in the preceding MOF class definition, which is defined
657 as follows:

```
658 [ ...
659   ArrayType ( "Indexed" ) ]
660 string StatusDescriptions[];
```

661 EXAMPLE 2: This array is defined as an element of the following complex type:

```
662 <xs:element name="StatusDescriptions" type="cim:cimString" nillable="true"/>
```

663 EXAMPLE 3: This array property, consisting of the following entries, appears in an XML instance document as
664 follows:

```
665 <EX_BaseComponent>
666   ...
667   <StatusDescriptions>SomeStatusDescription</StatusDescriptions>
668   <StatusDescriptions>AnotherStatusDescription</StatusDescriptions>
669   <StatusDescriptions>AThirdStatusDescription</StatusDescriptions>
670   ...
671 </EX_BaseComponent>
```

672 EXAMPLE 4: The MOF may also specify qualifiers that apply to each element in the array (for example, the
673 maximum length of each element).

674 NOTE 3: This example is not illustrated in the example class used in this specification.

```
675 [ ...
676   MaxLen ( 64 ) ]
677 string StatusDescriptions[];
```

678 EXAMPLE 5: In the following example, this restriction is defined on the `StatusDescriptions` element using an
679 anonymous complex type definition. The restriction base is the datatype that would otherwise have been assigned to
680 the element itself. See 11.2 for more information about applying qualifiers as restrictions.

```
681 <xs:element name="StatusDescriptions" nillable="true">
682   <xs:complexType>
683     <xs:restriction base="cim:cimString">
```

```

684     <xs:maxLength value="64"/>
685     <xs:anyAttribute namespace="##any" processContents="lax"/>
686   </xs:restriction>
687 </xs:complexType>
688 </xs:element>

```

689 9.2.2.2 Runtime Rules for Arrays

690 Specific rules for representing arrays in XML instance documents may apply at runtime:

- 691 • The position of each member of an array in its XML representation shall conform to
692 semantics regarding index and value defined by the `ArrayType` qualifier in the *CIM*
693 *Infrastructure Specification*, [DSP0004](#). Array index is inferred by the position of an element
694 relative to peer elements of the same name.
- 695 • Indexed arrays that include members that have a NULL value shall include each such
696 member in the XML representation of the array as an empty element with the `xsi:nil`
697 attribute for these elements set to the value `true`.

698 The `StatusDescriptions` array is an indexed array. If the second entry were deleted from the array,
699 the preceding example must be transmitted as follows:

```

700 <EX_BaseComponent>
701   ...
702   <StatusDescriptions>SomeStatusDescription</StatusDescriptions>
703   <StatusDescriptions xsi:nil="true"/>
704   <StatusDescriptions>AThirdStatusDescription</StatusDescriptions>
705   ...
706 </EX_BaseComponent>

```

707 9.2.3 Properties with a ValueMap Qualifier

708 This clause defines and illustrates the rules for mapping CIM properties with a `ValueMap` qualifier to
709 XSD.

710 The `ValueMap` qualifier shall be mapped as metadata fragments (see 11.1.2). The `ValueMap` qualifier
711 shall also be mapped in the XSD, according to the following rules.

712 Mapping of properties qualified with a `ValueMap` qualifier shall follow the general principles for mapping
713 properties in 9.2.1.1, with the following additions:

- 714 • If the `ValueMap` consists of only discrete values, the `ValueMap` shall be mapped to a
715 `<xs:restriction>` consisting of an enumeration as follows:
 - 716 – The base type of the restriction shall be the WS-CIM datatype corresponding to the CIM
717 datatype of the property (see 8).
 - 718 – Each discrete value of the `ValueMap` shall be mapped to a corresponding
719 `<xs:enumeration>` element within the restriction.
- 720 • If the `ValueMap` contains a value specifying a range, the whole `ValueMap` shall be mapped
721 to a `<xs:restriction>` consisting of a `<xs:union>` as follows:
 - 722 – The base type of the restriction shall be `cim:cimAnySimpleType` (see 8.4).
 - 723 – The elements of the `<xs:union>` shall be determined according to the following rules:
 - 724 • Discrete values shall be mapped to elements to an `<xs:restriction>` as
725 described in the first rule with the exception that the base type of the restriction shall
726 be the corresponding base XSD type of the CIM datatype of the property (see
727 clause 8);

- 728
- 729
- 730
- 731
- 732
- 733
- 734
- 735
- 736
- 737
- 738
- 739
- 740
- 741
- 742
- 743
- Bounded ranges (*m..n*) shall be mapped to an `<xs:restriction>` consisting of `<xs:minInclusive="m">` and `<xs:maxInclusive="n">` where the restriction base type shall be the corresponding base XSD type of the CIM datatype of the property;
 - Unbounded ranges open on the left (*...n*) shall be mapped to an `<xs:restriction>` consisting of `<xs:maxInclusive="n">` where the restriction base type shall be the corresponding base XSD type of the CIM datatype of the property;
 - Unbounded ranges open on the right (*m..*) shall be mapped to an `<xs:restriction>` consisting of `<xs:minInclusive="m">` where the restriction base type shall be the corresponding base XSD type of the CIM datatype of the property;
 - Open ranges (*..*) shall be mapped to an `<xs:union>` consisting of all discrete values and/or ranges that are unclaimed by the other values and ranges in the ValueMap by applying the preceding rules for constructing the elements of the `<xs:union>` recursively.

744 EXAMPLE 1: The following MOF fragment contains only discrete values for ValueMap:

```
745 [ ...
746     ValueMap { "OK", "Error", "Unknown" } ]
747 string HealthStatus;
```

748 The HealthStatus property is therefore mapped as follows:

```
749 <xs:element name="HealthStatus" nillable="true">
750     <xs:complexType>
751         <xs:simpleContent>
752             <xs:restriction base="cim:cimString">
753                 <xs:enumeration value="OK"/>
754                 <xs:enumeration value="Error"/>
755                 <xs:enumeration value="Unknown"/>
756                 <xs:anyAttribute namespace="##any" processContents="lax"/>
757             </xs:restriction>
758         </xs:simpleContent>
759     </xs:complexType>
760 </xs:element>
```

761 EXAMPLE 2: The following MOF fragment contains discrete values and bounded ranges for ValueMap:

```
762 [ ...
763 ValueMap { "0", "1", "2", "3..15999", "16000..65535" },
764     Values { "Unknown", "Other", "Not Applicable", "DMTF Reserved",
765         "Vendor Reserved" }]
766 uint16 PortType;
```

767 The PortType property is therefore mapped as follows:

```
768 <xs:element name="PortType" nillable="true">
769     <xs:complexType>
770         <xs:simpleContent>
771             <xs:restriction base="cim:cimAnySimpleType">
772                 <xs:simpleType>
773                     <xs:union>
774                         <xs:simpleType>
775                             <xs:restriction base="xs:unsignedShort">
776                                 <xs:enumeration value="0"/>
```

```

777         <xs:enumeration value="1"/>
778         <xs:enumeration value="2"/>
779     </xs:restriction>
780 </xs:simpleType>
781 <xs:simpleType>
782     <xs:restriction base="xs:unsignedShort">
783         <xs:minInclusive value="3"/>
784         <xs:maxInclusive value="15999"/>
785     </xs:restriction>
786 </xs:simpleType>
787 <xs:simpleType>
788     <xs:restriction base="xs:unsignedShort">
789         <xs:minInclusive value="16000"/>
790         <xs:maxInclusive value="65535"/>
791     </xs:restriction>
792 </xs:simpleType>
793 </xs:union>
794 </xs:simpleType>
795 <xs:anyAttribute namespace="##any" processContents="lax"/>
796 </xs:restriction>
797 </xs:simpleContent>
798 </xs:complexType>
799 </xs:element>

```

800 **EXAMPLE 3:** The following MOF fragment contains discrete values, an open range, and an unbounded range for
801 the ValueMap:

```

802 [ ...
803 ValueMap { "1", "2", "3", "4", "5", "6", "7", "..", "16000.." },
804     Values { "Other", "Create", "Delete", "Detect", "Read", "Write",
805         "Execute", "DMTF Reserved", "Vendor Reserved" }]
806 uint16 Activities;

```

807 The Activities property is therefore mapped as follows:

```

808 <xs:element name="Activities" nillable="true">
809     <xs:complexType>
810         <xs:simpleContent>
811             <xs:restriction base="cim:cimAnySimpleType">
812                 <xs:simpleType>
813                     <xs:union>
814                         <xs:simpleType>
815                             <xs:restriction base="xs:unsignedShort">
816                                 <xs:enumeration value="1"/>
817                                 <xs:enumeration value="2"/>
818                                 <xs:enumeration value="3"/>
819                                 <xs:enumeration value="4"/>
820                                 <xs:enumeration value="5"/>
821                                 <xs:enumeration value="6"/>
822                                 <xs:enumeration value="7"/>
823                             </xs:restriction>
824                         </xs:simpleType>
825                         <xs:simpleType>
826                             <xs:union>
827                                 <xs:simpleType>
828                                     <xs:restriction base="xs:unsignedShort">
829                                         <xs:enumeration value="0"/>
830                                     </xs:restriction>
831                                 </xs:simpleType>
832                             </xs:union>
833                         </xs:simpleType>
834                     </xs:union>
835                 </xs:restriction>
836             </xs:simpleContent>
837         </xs:complexType>
838     </xs:element>

```



```

833         <xs:restriction base="xs:unsignedShort">
834             <xs:minInclusive value="8"/>
835             <xs:maxInclusive value="15999"/>
836         </xs:restriction>
837     </xs:simpleType>
838 </xs:union>
839 </xs:simpleType>
840 <xs:simpleType>
841     <xs:restriction base="xs:unsignedShort">
842         <xs:minInclusive value="16000"/>
843     </xs:restriction>
844 </xs:simpleType>
845 </xs:union>
846 </xs:simpleType>
847 <xs:anyAttribute namespace="##any" processContents="lax"/>
848 </xs:restriction>
849 </xs:simpleContent>
850 </xs:complexType>
851 </xs:element>

```

852 9.2.4 Octetstring Properties

853 The `Octetstring` qualifier may be applied to either `uint8` arrays or `string` arrays. In `uint8` arrays,
 854 the property identifies only a single binary entity; in `string` arrays, each string in the array represents a
 855 different binary entity.

856 9.2.4.1 General Rules

857 The rules for representing properties that are octetstrings are as follows:

- 858 • A `uint8` array that is designated as an octetstring shall be mapped to a single XSD
 859 element of the type `cim:cimBase64Binary`. The rules for mapping properties defined in
 860 9.2.1.1 apply to this mapping.
- 861 • A `string` array that is designated as an octetstring shall be mapped to an array of type
 862 `cim:cimHexBinary`. The rules for mapping arrays defined in 9.2.2.1 apply to this
 863 mapping.
- 864 • The XML payload does not include the 0x or the length bytes for the string form of octet
 865 string. The `uint8` form does not include the length bytes.

866 EXAMPLE 1: The following `uint8` array is designated as an octetstring:

```

867 [...
868     Description ("The DER-encoded raw public key. " ),
869     OctetString ]
870 uint8 PublicKey[];

```

871 It would be represented by the following XSD:

```

872 <xs:element name="PublicKey" type="cim:cimBase64Binary" nillable="true"/>

```

873 It would be represented in an XML instance document by entries such as the following:

```

874 <PublicKey>AAAAExEiM0RVZneImaq7zN3u/w==</PublicKey>

```

875 EXAMPLE 2: The following CIM `string` array is designated as an octetstring:

```

876 [...
877     Description ("A CRL, or CertificateRevocationList, is a list of certificates which the "
878         "CertificateAuthority has revoked and which are not yet expired. " ),
879     Octetstring]
880 string CRL[];

```

881 It would be represented by the following XSD:

```
882 <xs:element name="CRL" type="cim:cimHexBinary" nillable="true"/>
```

883 It would be represented in an XML instance document by entries such as the following:

```
884 <CRL>76C8A4...</CRL>
```

```
885 <CRL>75D4E1...</CRL>
```

```
886 <CRL>B1C335...</CRL>
```

887 9.2.4.2 Runtime Value Conversion Rules

888 This clause defines the normative rules for the runtime conversion rules for values of octetstring
889 properties.

890 The hex format for the `string` array variant of octetstrings is used to avoid additional conversion steps in
891 the XML protocol layer, which would need to convert the hex encoding generated by the CIM provider to
892 binary and then convert that binary to base64. The values in the preceding examples (see 9.2.4.1) are
893 obtained by applying the following runtime value conversion rules:

- 894 • A `uint8` array that is designated as an octetstring shall be converted to its corresponding
895 representation in `base64Binary` such that the ordered set of array elements is
896 concatenated into a binary multi-octet string, which is converted to base64 encoding. This
897 encoding represents the `base64Binary` value. The order of the unsigned 8-bit integer array
898 shall be preserved when mapped to the characters of the XML value.
- 899 • A `string` array that is designated as an octetstring shall be converted to its corresponding
900 representation in `hexBinary` such that for each string array element, one `hexBinary` element
901 is created, with the unchanged value of the string array element.

902 9.2.5 EmbeddedObject and EmbeddedInstance Properties

903 `EmbeddedObject` and `EmbeddedInstance` qualifiers apply to string properties whose values are
904 complete encodings of the data of an instance or class definition. An `EmbeddedObject` property may
905 contain either the encoding of an instance's data or a class definition; an `EmbeddedInstance` property
906 contains only the encoding of an instance's data.

907 9.2.5.1 General Rules

908 The rule for represented string properties that are designated as `EmbeddedObjects` or
909 `EmbeddedInstances` is as follows:

910 The general rules for mapping properties in 9.2.1.1 apply to properties that contain embedded
911 objects or instances, with the following exception: The property shall be converted to an element of
912 type `xs:anyType`.

913 EXAMPLE: The following MOF fragment defines a string property that contains an embedded object:

```
914 [ ...  
915     EmbeddedObject ( "...") ]  
916 string TheObject;
```

917 It would have the following XSD representation:

```
918 <xs:element name="TheObject" type="xs:anyType" nillable="true"/>
```

919 9.2.5.2 Runtime Value Conversion Rules

920 Runtime conversion of actual values of an `EmbeddedInstance` or `EmbeddedObject` property requires
921 different algorithms depending on the representation in the property. For example, the encoding of the
922 instance or class may be provided through MOF or CIM-XML encoding.

923 This clause defines the normative rules for the runtime conversion of embedded instances and embedded
924 objects, as follows:

- 925 • If the CIM property that is qualified by an `EmbeddedInstance` or an `EmbeddedObject`
926 qualifier contains an instance, then
 - 927 – The property value shall be converted to an XML instance representation as if the XSD
928 type of the property was the actual XSD type of the class of the instance.
 - 929 – The property element shall contain an `xsi:type` attribute with the XSD type of the class
930 of the instance (see 9.3.1).
- 931 • If the CIM property that is qualified by an `EmbeddedObject` qualifier contains a class
932 definition, the property value shall be converted to the XML Schema of that class. See 9.6
933 for the normative rules for representing CIM instances.

934 EXAMPLE: The following class definition in MOF embeds an instance of `CIM_Part` in `CIM_Component`:

```
935 class CIM_Part {
936     string Label;
937     int PartNo;
938 };
939 class CIM_Component {
940     [Key]
941     string ID;
942     [EmbeddedInstance("CIM_Part")]
943     string Part;
944 };
```

945 Given an embedded instance of `CIM_Part` with `Label="Front Panel"` and `PartNo="19932"`, the
946 following is a valid instance representation in the runtime XML instance document:

```
947 <CIM_Component xmlns="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_Component"
948     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
949     <ID>ua123</ID>
950     <Part xmlns:e="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_Part"
951         xsi:type="e:CIM_Part_Type">
952         <e:Label>Front Panel</e:Label>
953         <e:PartNo>19932</e:PartNo>
954     </Part>
955 </CIM_Component>
```

956 9.3 Class Structure

957 This clause describes the XSD representation of CIM classes. The intended scope is all classes defined
958 in CIM, including associations, indications, and exceptions. Associations, indications, and exceptions are
959 distinguished by having an effective `Association`, `Indication` or `Exception` qualifier, respectively.

960 NOTE: These qualifiers have standard mappings to metadata fragments for these classes (see 11.1.1).

961 9.3.1 General Rules

962 CIM classes are represented in the XML Schema according to the following rules:

- 963 • The structure of the CIM class shall be mapped to an XML global complex type definition.
964 The definition of this complex type shall comply with the following rules:
 - 965 – The name of this type shall match that of the CIM class name, including the CIM schema
966 name, with the suffix `_Type`.

- 967 – The complex type shall consist of an `<xs:sequence>` that contains the set of elements
 968 referring to the GEDs that define the properties of the class (see 9.2). These elements
 969 have the following form:

970

```
<xs:element ref=' QName ' . ' Attributes '/>
```

971 Where:

- 972 • QName is the QName of the GED that represents the target property.
 973 • Attributes represents any required attributes (such as `minOccurs` and `maxOccurs`).

974 Elements that belong to the class complex type should be ordered by Unicode code point
 975 (binary) order of their CIM property name identifiers. (See ANNEX C for information about
 976 collation.)

- 977 • In existing service implementations, the collation sequence of property name
 978 identifiers should be in Unicode code point (binary) order.
 979 • In existing service implementations, the collation sequence of property name
 980 identifiers may be according to the Unicode Collation Algorithm (UCA) with its default
 981 settings. This algorithm is deprecated and should not be used in future service
 982 implementations.
 983 • In DMTF XML schema representations of CIM classes, and in new service
 984 implementations, the collation sequence of property name identifiers shall be in code
 985 point (binary) order.

986 The following rules apply to specifying the multiplicity of these elements:

- 987 • All elements that do not represent array properties shall have `minOccurs="0"` except for
 988 elements that correspond to properties that are designated with a `Key Or Required`
 989 qualifier with an effective value of `true`. Elements that do not represent arrays and
 990 represent key and required properties shall have `minOccurs="1"`. Because 1 is the
 991 default value for `minOccurs`, it does not need to be explicitly expressed.
 992 • All elements that represent array properties shall have `minOccurs="0"`. If the array size
 993 is specified in the CIM definition, the array property shall have `maxOccurs="array`
 994 `size"`. If the array size is not specified, the array property shall have
 995 `maxOccurs="unbounded"`.
 996 • All elements except arrays shall have `maxOccurs="1"`. Because 1 is the default value for
 997 `maxOccurs`, it does not need to be explicitly expressed.
 998 • Array properties (see 9.2.1.1) shall have a multiplicity that corresponds to the specification
 999 of their size in CIM. A bounded array in CIM shall be specified with a `maxOccurs` equal to
 1000 the size of the array. If no size is specified in CIM, the schema element shall be specified
 1001 with `minOccurs="0"` and `maxOccurs="unbounded"`.
 1002 • The schema of the CIM class shall support an open schema. Open schema means
 1003 different protocols are able to add protocol-specific elements to instance documents.

- 1004 – To allow Open Content, the final element in the sequence shall be as follows:

1005

```
<xs:any namespace="##other" processContents="lax" minOccurs="0"
```


 1006

```
maxOccurs="unbounded" />
```

- 1007 – To allow attributes to be added to the element that represents the CIM class, following the
 1008 sequence, the complex type shall allow any attribute to be added to the class with an
 1009 `xs:anyAttribute` element, as follows:

1010

```
<xs:anyAttribute namespace="##any" processContent="lax" />
```

- 1011 • The class itself shall be represented by a GED of the type defined in the preceding rule.
 1012 The name of this element shall be the name of the CIM class including its CIM schema
 1013 name.

1014 **EXAMPLE:** The class defined in 9.2.1 has the following mapping as an XSD class definition:

```

1015 <xs:complexType name="EX_BaseComponent_Type">
1016   <xs:sequence>
1017     <xs:element ref="class:HealthStatus" minOccurs="0"/>
1018     <xs:element ref="class:InstallDate" minOccurs="0"/>
1019     <xs:element ref="class:Name"/>
1020     <xs:element ref="class:StatusDescriptions" minOccurs="0" maxOccurs="unbounded"/>
1021     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1022       maxOccurs="unbounded"/>
1023   </xs:sequence>
1024   <xs:anyAttribute namespace="##any" processContent="lax"/>
1025 </xs:complexType>
1026 <xs:element name="EX_BaseComponent" type="class:EX_BaseComponent_Type"/>

```

1027 The complete mapping of this class is provided in B.2.1.

1028 9.3.2 Runtime Property Value Rules

1029 Runtime inclusion of property values for instance documents based on the XML Schema for CIM classes
 1030 is defined by the following rules:

- 1031 • For "get" operations, CIM service implementations returning the class's GEDs may omit
 1032 schema-optional (`minOccurs="0"`) properties that have NULL values from responses.
 1033 Clients are to interpret such omitted properties as having NULL values for those properties.
- 1034 • Empty arrays (arrays with no members) shall not be included in responses to "get"
 1035 requests. If the array is required, clients shall interpret the absence of all elements for the
 1036 array to mean that the array is empty (no members). If the array is not required, clients
 1037 shall interpret the absence of all elements for the array to mean that the array is either
 1038 empty or NULL.
- 1039 • A WS-CIM server shall return all current elements of an array.
- 1040 • For "set" operations using the class's GEDs, clients may omit schema-optional
 1041 (`minOccurs="0"`) properties. Service interpretation of the absence of such properties is
 1042 protocol dependent.
- 1043 • An instance document conformant to this specification shall include any elements from a
 1044 foreign namespace at the end of the sorted list of CIM class elements. Elements from a
 1045 foreign namespace may appear in any order.
- 1046 • If a `Version` qualifier is associated with the CIM class, the `cim:Version` attribute may
 1047 be applied to the element that represents the class in an XML instance document. Use of
 1048 the `cim:Version` attribute shall conform to the following rule:

1049 If the attribute is present, its value shall be assigned as the value of the `Version` qualifier that
 1050 is associated with the CIM class, in the XML instance. If the CIM class does not have the
 1051 `Version` qualifier associated with it, this attribute shall be omitted.

1052 The MOF typically contains the `Version` qualifier, which specifies the latest version of the CIM class in
 1053 CIM:

```

1054 [ ... Version ( "2.10.2" ) ]

```

```

1055 class EX_SomeClass {
1056   ...
1057 }

```

1058 The class may be represented in an instance document as follows:

```

1059 <EX_SomeClass cim:Version="2.10.2">
1060   ...
1061 </EX_SomeClass>

```

1062 9.4 Class Inheritance

1063 CIM inheritance is not modeled in the XML Schema of classes or within XML instances.

1064 Class inheritance is governed by the following rules:

- 1065 • Besides including the GEDs for the properties defined in a class (see 9.2.1.1), the
1066 namespace for a class shall also include the GEDs for properties inherited from its
1067 superclasses. The class type definition shall contain references to GEDs for all properties
1068 defined in and inherited into the class according the rules in 9.3.1.
- 1069 • In general, classes inherit all properties specified in or inherited by their superclasses along
1070 with all qualifiers that are specified as `ToSubClass`. However, properties with the same
1071 name may be encountered within an inheritance chain. The `Override` qualifier
1072 determines special behaviors that shall be observed by conversion algorithms when
1073 encountering properties with duplicate names in the inheritance chain. The following rules
1074 govern the mapping of properties with duplicate names:
 - 1075 – When multiple properties in the inheritance chain that have the same name are not
1076 overridden (that is, the effective value of the `Override` qualifier throughout the inheritance
1077 chain is `NULL`), the property and its qualifiers in the most derived class shall be mapped.
1078 All other duplicate named properties shall not be mapped.
 - 1079 – When a property in a derived class overrides another property (of the same name and
1080 type) in a superclass, the property in the most derived class shall be mapped, including all
1081 qualifiers inherited from the overridden property. The overridden property shall not be
1082 mapped.
- 1083 • The inheritance of qualifiers that pertain to properties shall comply with the inheritance
1084 rules regarding qualifiers in B.3.
- 1085 • The definition of a derived class shall follow all other rules for constructing classes as
1086 defined in 9.3.1.

1087 NOTE: The metadata fragments for a property shall include any inherited qualifiers, subject to the qualifier inheritance
1088 rules defined in 11.3. For more information about metadata fragments, see clause 11.

1089 Inheritance rests on the same type of examples presented in 9.2 and 9.3. The only addition is that the
1090 properties inherited from a class's superclasses are included in the GEDs and class complex type
1091 definition. For a complete example of inheritance, see B.2.2.

1092 9.5 Method Parameters

1093 The invocation of a CIM extrinsic method is represented by two separate messages:

- 1094 • the request input message, which represents the invocation of the method and the set of
1095 input parameters
- 1096 • the response output message, which represents the output parameters and the method
1097 return value in the successful case

1098 This clause specifies the XML Schema for these elements. These elements are then included as parts in
1099 the WSDL input and output messages (see 10.1).

1100 9.5.1 General Rules

1101 The rules in this clause apply to mapping method input and output parameters and method return values.

1102 The GED used for the request input message is called the *input message GED*. The GED used for the
1103 response output message is called the *output message GED*. The following rules specify the definition of
1104 these GEDs:

- 1105 • The class namespace of the CIM class being mapped shall contain the input and output
1106 message GEDs of all methods owned by the class and inherited from the superclasses.
1107 See 9.5.2, which defines class ownership of methods inherited from superclasses.
- 1108 • The names of these GEDs are defined by the following rules:
 - 1109 – The *name* of the input message GED shall be the name of the CIM method with `_INPUT`
1110 appended.
 - 1111 – The *name* of the output message GED shall be the name of the CIM method with `_OUTPUT`
1112 appended.
- 1113 • Each GED shall be defined as a complex type containing an `xs:sequence` of in-line
1114 elements that represent the respective input or output parameters and return value of the
1115 CIM method as immediate children. This structure is further defined in the rest of this
1116 clause.

1117 The following rules define the mapping of individual input and output parameters and the return value of
1118 the CIM method, and the structure of the complex type used for defining the GEDs:

- 1119 • Input and output parameters of a CIM method shall be mapped to elements with the same
1120 name as the corresponding parameter name. The following rules define the features of
1121 these elements:
 - 1122 – The type of an element that represents an input or output parameter shall be mapped as
1123 defined in clause 8.
 - 1124 – Parameters that are not qualified with a `Required` qualifier shall be mapped to elements
1125 that contain the `nillable="true"` attribute.
 - 1126 • The complex type used to define the type of the input message GED shall contain all and
1127 only those elements that correspond to method parameters that have their `In` qualifier
1128 effectively defined as `true`. The sequence of these elements in the complex type shall
1129 correspond to the sequence of the input parameters in the CIM definition of the method.
- 1130 If the method has no input parameters, the complex type used in the GED shall be empty (that
1131 is, `<xs:complexType/>`).
- 1132 • The complex type used to define the type of the output message GED shall contain all
1133 elements that correspond to method parameters that have their `Out` qualifier effectively
1134 defined as `true`. The sequence of these elements in the complex type shall correspond to
1135 the sequence of the output parameters in the CIM definition of the method.
 - 1136 • The complex type used to define the output message GED shall contain an element of
1137 `name="ReturnValue"` as the final element in its sequence. The following rules govern
1138 the structure of this element:
 - 1139 – The XSD type of this element shall be mapped from the CIM method type in compliance
1140 with the datatype conversion defined in clause 8.

- 1141 – The element shall include the attribute `nillable="true"`. (See the following note.)
- 1142 • Parameters and return values may be defined in CIM with `ValueMap` qualifiers. Mapping
- 1143 these `ValueMaps` to metadata fragments is required (see 11.1). In addition, a `ValueMap`
- 1144 shall be mapped to an enumeration/union associated with the mapped parameter or return
- 1145 value in the XSD (see 11.2). The mapping shall conform to the rules for mapping
- 1146 `ValueMaps` described in 9.2.3.

1147 Notes on the preceding rules:

- 1148 • A parameter shall occur in both the complex types used to define the input and output
- 1149 message GEDs if it has both the `In` and `Out` qualifiers effectively defined as `true`.

1150 [DSP0004](#) allows NULL as the default return value for all methods. Thus, this specification must allow for

1151 the possibility that the return value of any method may be NULL.

1152 9.5.2 Inheritance of Methods

1153 Classes may inherit some of the methods that they own. In general, classes inherit all methods specified

1154 in or inherited by their superclasses, along with all qualifiers that are specified as `ToSubClass`. The

1155 `Override` qualifier, however, determines special behavioral considerations on the part of conversion

1156 algorithms. The following rules govern the inheritance of methods under conditions of override:

- 1157 • When multiple methods in the inheritance chain that have the same name are not
- 1158 overridden, the method in the most derived class shall be mapped. Any other duplicate, but
- 1159 not overridden, methods shall not be mapped.
- 1160 • When a method in a derived class overrides another method (of the same name and
- 1161 signature) in a superclass, the method in the most derived class shall be mapped,
- 1162 including all qualifiers inherited from the overridden methods. The overridden method shall
- 1163 not be mapped.
- 1164 • The inheritance of qualifiers pertaining to methods shall comply with the inheritance rules
- 1165 regarding qualifiers in B.3.

1166 EXAMPLE: As an example of the preceding rules, consider the following MOF method definition extracted from the

1167 example in B.1.2. (See B.2.2 for the complete example.)

```
1168 class EX_DerivedComponent
1169 {
1170 ...
1171 uint32 RequestStateChange([IN] uint16 RequestedState, [OUT] [IN(False)] CIM_SomeClass REF
1172 ResultClass, [IN] datetime TimeoutPeriod);
1173 };
```

1174 The input parameters, designated in the MOF by the `In` qualifier, would be mapped as follows:

```
1175 <xs:element name="RequestStateChange_INPUT">
1176 <xs:complexType>
1177 <xs:sequence>
1178 <xs:element name="RequestedState" type="cim:cimUnsignedShort"
1179 nillable="true"/>
1180 <xs:element name="TimeoutPeriod" type="cim:cimDateTime"
1181 nillable="true"/>
1182 </xs:sequence>
1183 </xs:complexType>
1184 </xs:element>
```

1185 The output parameters, designated in the MOF by the `Out` qualifier, would be mapped in the following

1186 way. Note that the complex type includes an element that represents the return value of the CIM method.


```

1187 <xs:element name="RequestStateChange_OUTPUT">
1188 <xs:complexType>
1189   <xs:sequence>
1190     <xs:element name="ResultClass" type="cim:cimReference"
1191       nillable="true"/>
1192     <xs:element name="ReturnValue" type="cim:cimUnsignedInt"
1193       nillable="true"/>
1194   </xs:sequence>
1195 </xs:complexType>
1196 </xs:element>

```

1197 9.5.3 Parameters and Return Values with ValueMaps

1198 Input and output parameters and return values of the method may be specified with a `ValueMap` qualifier.
 1199 The `ValueMap` shall be mapped to the XSD element that represents the parameter or return value.

1200 The `RequestedState` input parameter must be defined as an enumeration in accordance with its
 1201 `ValueMap` (see B.1.2 for this `ValueMap`):

```

1202 <xs:element name="RequestedState" nillable="true">
1203 <xs:complexType>
1204   <xs:simpleContent>
1205     <xs:restriction base="cim:cimUnsignedShort">
1206       <xs:enumeration value="2"/>
1207       <xs:enumeration value="3"/>
1208       <xs:enumeration value="4"/>
1209       <xs:anyAttribute namespace="##any" processContents="lax"/>
1210     </xs:restriction>
1211   </xs:simpleContent>
1212 </xs:complexType>
1213 </xs:element>

```

1214 9.6 CIM Instances

1215 This clause describes the representation of CIM instances. The intended scope is all representations of
 1216 CIM instances used in any protocols.

1217 CIM instances are represented according to the following rules:

- 1218 • Representations of CIM instances shall be XML instance documents that conform to the
 1219 XSD schema for their CIM creation class, as defined in clause 9.
- 1220 • The class namespace used within an instance document shall be a namespace URI and it
 1221 shall be defined as follows:

1222 ['http://schemas.dmtf.org/wbem/wscim/ wscim-major-version /cim-schema/ cim-schema-major-
 version /' cim-class-schema '_' cim-class-name](http://schemas.dmtf.org/wbem/wscim/ wscim-major-version /cim-schema/ cim-schema-major-

 1223 version /' cim-class-schema '_' cim-class-name)

1224 Where:

- 1225 – [wscim-major-version](#) is the major version number of this specification. Note that this
 1226 version number changes only if there are incompatible changes in the specification.
- 1227 – [cim-schema-major-version](#) is the major version number of the CIM schema version to
 1228 which the class being converted belongs. Note that this version number changes only if
 1229 there are incompatible changes in the CIM schema.
- 1230 • The location of the specific schema used to construct the instance should be declared by
 1231 use of the `xsi:schemaLocation` attribute where the namespace URI and the specific class

1232 schema document URI location are combined as the value of the attribute, separated by
 1233 whitespace. This provides a means for a recipient of the instance to determine which
 1234 version of CIM defines the structure of this instance.

1235 For example:

1236 If the instance document is constructed under CIM schema version 2.11.0 and WS-CIM
 1237 mapping specification 1.3.0, the following `xsi:schemaLocation` attribute should be specified in
 1238 the instance document (where *ClassName* is the name of the CIM class of the instance):

```
1239 xsi:schemaLocation="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/ClassName  

  1240 http://schemas.dmtf.org/wbem/wscim/1.3.0/cim-schema/2.11.0/ClassName.xsd"
```

1241 If the URI in the attribute value can be de-referenced, then strict XML schema validation can be achieved.

1242 9.7 Superclass Class Representation

1243 The CIM Schema defines CIM classes in subclass / superclass relationships (hierarchies). For example,
 1244 MOF reflects this structure in the definition of the class. A MOF statement of the following form defines
 1245 `CIM_ClassA` as a subclass of the superclass `CIM_ClassB`:

```
1246 class CIM_ClassA : CIM_ClassB
```

1247 A MOF statement of the following form defines `CIM_ClassC` as a top-level class with no superclass:

```
1248 class CIM_ClassC
```

1249 Some management protocols may require a representation of the subclass/superclass hierarchy of a
 1250 class as a value of an XML element in instance documents. The XSD mechanism by which to support
 1251 representing this structure as a value of an XML element is provided in a separate schema that may be
 1252 imported by protocols that require this capability for their instance documents.

1253 The immediate subclass/superclass relationship of a class shall be mapped to a single GED in the
 1254 classhierarchy namespace according to the following rules:

- 1255 • Elements that describe the subclass / superclass relationships shall be placed in a
 1256 separate schema from the WS-CIM class schema. For subclass / superclass relationships
 1257 defined in the CIM Schema, the schema shall be named as follows:

```
1258 'http://schemas.dmtf.org/wbem/wscim/' wscim-major-version 'cim-schema/' cim-schema-  

  1259 major-version 'classhierarchy'
```

1260 Where:

- 1261 – `wscim-major-version` is the major version number of this specification. Note that this
 1262 version number changes only if there are incompatible changes in the specification.
- 1263 – `cim-schema-major-version` is the major version number of the CIM schema version to
 1264 which the class being converted belongs. Note that this version number changes only if
 1265 there are incompatible changes in the CIM schema.

1266 This schema shall import the `http://schemas.dmtf.org/wbem/wscim/n/classhiertype` namespace, where '*n*'
 1267 represents the version number of this namespace.

1268 The class being mapped shall be represented by a GED whose name is derived from the name of the
 1269 CIM class, appended with `"_Class"`. This element shall be defined by an anonymous complex type that
 1270 follows one of the following patterns:

- 1271 • The WS-CIM schema of a top-level class, `XXX_ClassC`, shall define the `_Class` element
 1272 as a restriction of the `ctype:ClassHierarchyType`. The following pattern illustrates this
 1273 rule:

```

1274 <xs:element name="XXX_ClassC_Class">
1275   <xs:complexType>
1276     <xs:complexContent>
1277       <xs:restriction base="ctype:ClassHierarchyType" />
1278     </xs:complexContent>
1279   </xs:complexType>
1280 </xs:element>

```

- The WS-CIM schema of a subclass, XXX_ClassA, that is derived by a superclass, XXX_ClassB, shall restrict the `ctype:ClassHierarchyType` to contain a single element that references the corresponding `_Class` GED of the superclass. The following pattern illustrates this rule:

```

1285 <xs:element name="XXX_ClassA_Class">
1286   <xs:complexType>
1287     <xs:complexContent>
1288       <xs:restriction base="ctype:ClassHierarchyType">
1289         <xs:sequence>
1290           <xs:element ref="chier:XXX_ClassB_Class" />
1291         </xs:sequence>
1292       </xs:restriction>
1293     </xs:complexContent>
1294   </xs:complexType>
1295 </xs:element>

```

1296 See B.2.4 for an example classhierarchy schema.

1297 10 CIM Methods to WSDL Mappings

1298 This clause defines the structures that are necessary to define the messages and operation structures
 1299 required for mapping a CIM method to WSDL.

1300 10.1 Defining WSDL Message Structures

1301 This clause provides the rules for creating WSDL message structures.

1302 The rules that govern the creation of WSDL message structures for a method are as follows:

- The name of the WSDL input message should be the name of the CIM method plus `_InputMessage`. The following rules specify the structure of this element:
 - The name of the `wsdl:part` element should be "body".
 - The `element` attribute of the `wsdl:part` element shall specify the QName of the input message GED for the CIM method (see 9.5).
- The name of the WSDL output message should be the name of the CIM method plus `_OutputMessage`. The following rules specify the structure of this element:
 - The name of the `wsdl:part` element should be "body".
 - The `element` attribute of the `wsdl:part` element shall specify the QName of the output message GED defined for the CIM method (see 9.5).

1313 **EXAMPLE:** The `wsdl:message` elements for the `RequestStateChange` CIM method (see 9.5) would
 1314 be specified in the WSDL document as follows. The `wsdl:message` intended for input to the WSDL
 1315 operation would be as follows (where the "class:" namespace prefix represents the namespace of the
 1316 class whose interface is being exposed through this WSDL):

```

1317 <wsdl:message name="RequestStateChange_InputMessage">
1318   <wsdl:part name="body"
1319     element="class:RequestStateChange_INPUT" />
1320 </wsdl:message>

```

1321 See B.3 for an example that shows the complete specification of the WSDL messages for this operation.

1322 10.2 Defining WSDL Operation Structures

1323 This specification defines *only* the structure of WSDL `portType` operations.

1324 The rules governing the structure of the WSDL operations used to invoke CIM methods are as follows:

- 1325 • The name attribute of the `wsdl:operation` element shall be the name of the CIM
1326 method.
- 1327 • The name attributes of the `wsdl:input` and `wsdl:output` child elements should be the
1328 name of the `wsdl:messages` that are referenced by these elements.
- 1329 • The message attributes of the `wsdl:input` and `wsdl:output` elements shall specify the
1330 QName of input and output message elements defined in 10.1.

1331 EXAMPLE: The `RequestStateChange` CIM method (see 9.5) is defined as follows:

```

1332 <wsdl:definitions
1333   ...
1334   xmlns:thisWSDL="http://. . .wsdl"
1335   ...>
1336 <wsdl:operation name="RequestStateChange">
1337   <wsdl:input name="RequestStateChange_InputMessage"
1338     message="thisWSDL:RequestStateChange_InputMessage" />
1339   <wsdl:output name="RequestStateChange_OutputMessage"
1340     message="thisWSDL:RequestStateChange_OutputMessage" />
1341 </wsdl:operation>
1342 </wsdl:definitions>

```

1343 This definition should be included in the `wsdl:portType` section of a WSDL document of a service that
1344 makes the CIM `RequestStateChange` method available to clients.

1345 10.3 Defining `wsa:Actions`

1346 The Addressing specifications ([Web Services Addressing \(WS-Addressing\) 1.0 – Core](#) and [DSP0226](#),
1347 "Management Addressing" clause) define the information model and syntax for the abstract messaging
1348 property Action. This property is defined as an IRI ([RFC 3987](#)), which identifies the semantics implied by
1349 a message (input, output, or fault). For the purposes of this specification, the Action property is restricted
1350 to a URI ([RFC 3986](#)) (a sequence of characters from a limited subset of the repertoire of US-ASCII
1351 characters).

1352 The details of how the action URI is specified on description artifacts are left to the specific protocol-
1353 binding specifications. Action URIs for faults are always left to the protocol-binding specifications.

1354 The rules for constructing WSA action URIs for input and output operation elements are as follows:

- 1355 • The action URI for an input message shall have the following form:

1356 class-namespace-name '/' input-name

1357 Where:

- 1358 – class-namespace-name is the full namespace name of the class being mapped as defined
1359 in 9.1.
- 1360 – input-name is the name of the CIM method.
- 1361 • The action URI for an output message shall have the following form:

1362 class-namespace-name '/' output-name 'Response'

1363 Where:

- 1364 – class-namespace-name is the full namespace name of the class being mapped as defined
1365 in 9.1.
- 1366 – output-name is the name of the output CIM method.

1367 EXAMPLE: The action URI for the input message of the `RequestStateChange` method is as follows:

```
1368 wsa:Action="http://schemas.dmtf.org/wbem/ws-cim/1/cim-schema/2/EX_DerivedComponent/  
1369 RequestStateChange"
```

1370 The action URI for the output message of the `RequestStateChange` method is as follows:

```
1371 wsa:Action="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/EX_DerivedComponent/  
1372 RequestStateChangeResponse"
```

1373 See B.3 for an example that shows a complete mapping of the `RequestStateChange` method.

1374 11 Qualifier Mappings

1375 This clause defines the mapping of qualifiers to metadata fragments. The definition of the container for
1376 metadata fragments is left to the specific protocols to specify.

1377 11.1 General Format

1378 Rules for mapping qualifiers fall into two categories:

- 1379 • rules that describe the type definitions of qualifiers
- 1380 • rules that describe the XSD elements to which qualifiers are mapped

1381 This specification provides type definitions for the qualifier types used in CIM in `common.xsd` and
1382 mappings for all CIM qualifiers in the `qualifiers.xsd` (Annex A.2). It is expected that user-defined qualifiers
1383 follow the mapping rules for mapping qualifiers described in the following clauses.

1384 In addition to the mapping rules, user-defined qualifier mappings are governed by the following rules:

- 1385 • User-defined qualifiers shall not use any qualifier namespace defined in this specification.

1386 The qualifier types (types with names of the qualifier *Type*) defined in `common.xsd` should be used to
1387 define user-defined qualifiers. In the majority of cases, it is not necessary to create a new type definition
1388 to define a qualifier.

1389 11.1.1 Single-Valued Qualifiers

1390 This clause describes the rules for mapping single-valued qualifiers.

1391 Single-valued qualifiers that have an effective value that matches their default value shall not be mapped
1392 to a metadata fragment.

1393 The rules for mapping single-valued qualifiers that have an effective value that is not their global default
1394 value are as follows:

- 1395 • The rules for defining the type of a single-valued qualifier are as follows:
- 1396 – Single-valued qualifiers shall be mapped to a complex type that is an extension of a simple
1397 content.
- 1398 – The base type of this complex type shall correspond to the type of qualifier according to the
1399 datatype conversion rules defined in clause 8.
- 1400 – The complex type shall extend the base type with a Boolean attribute of the name
1401 *qualifier*. This attribute is defined in the *cim* namespace. This attribute shall be
1402 specified with the XML Schema attribute *use="required"*.
- 1403 • The rules for mapping a single-valued qualifier are as follows:
- 1404 – The qualifier shall be mapped to a GED whose type corresponds to the datatype type of
1405 the qualifier and which has been defined by the preceding rule. The name of the element
1406 shall be the name of the qualifier.
- 1407 – The value of the *cim:qualifier* attribute in the mapped metadata fragment shall be *true*
1408 in all mappings.
- 1409 – Single-valued qualifiers implicitly have the multiplicity *minOccurs="0" maxOccurs="1"*.
1410 Protocols should provide a mechanism by which to express this multiplicity in the schema
1411 of the document that contains generated metadata fragments.
- 1412 • The format of a schema element for defining a single-valued qualifier is as follows:
- 1413

```
'<xs:element name="" cim:qualifier-name "" type="" qualifier-type ""/>'
```
- 1414 Where:
- 1415 – *qualifier-type* is typically one of the *qualifierType* types defined in *common.xsd* (but may be
1416 a user-defined type that complies with the mapping rules).

1417 *cim:qualifier-name* is the name of the qualifier, qualified by its namespace prefix.

1418 EXAMPLE: A generalized example of the mapping of a single-valued qualifier is as follows:

```
1419 <ns:QualifierName cim:qualifier="true">
1420    value
1421 </ns:QualifierName>
```

1422 Where:

- 1423 – *QualifierName* is the name of the qualifier.
- 1424 – *ns* is the namespace prefix referencing the namespace in which this qualifier is defined.
- 1425 – *value* is the string representation of the qualifier value.

1426 For example, the mapping of the CIM qualifier *Abstract* is as follows:

```
1427 <cimQ:Abstract cim:qualifier="true">true</cimQ:Abstract>
```

1428 11.1.2 Multi-Valued Qualifiers

1429 This clause describes the rules for mapping multi-valued qualifiers.

1430 Multi-valued qualifiers are qualifiers that are arrays. Multi-valued qualifiers that have an effective value
1431 that matches their default value shall not be mapped to a metadata fragment.

1432 The rules for mapping multi-valued qualifiers that have an effective value that is not their global default
 1433 value are as follows:

- 1434 • The rules for defining the type of a multi-valued qualifier are as follows:
 - 1435 – Multi-valued qualifiers shall be mapped to a complex type that is an extension of a complex
 1436 content.
 - 1437 – The base type of this complex type shall correspond to the single-valued qualifier *Type* of
 1438 the member elements from which the qualifier array is constructed.
- 1439 • The rules for mapping a multi-valued qualifier are as follows:
 - 1440 – The qualifier shall be mapped to a GED whose type corresponds to the datatype type of
 1441 the qualifier and which has been defined by the preceding rule. The *name* of the element
 1442 shall be the same as that of the qualifier itself.
 - 1443 – The value of the `cim:qualifier` attribute in the metadata fragments shall be `true` in all
 1444 mappings.
 - 1445 – Multi-valued qualifiers implicitly have the default multiplicity `minOccurs="0"`
 1446 `maxOccurs="unbounded"`. Qualifier declarations may explicitly set different bounds on
 1447 an array qualifier. Protocols should provide a mechanism by which to express the size of
 1448 an array qualifier in the schema of the document that contains generated metadata
 1449 fragments. The `minOccurs` and `maxOccurs` values shall correspond either to the default
 1450 values if no qualifier array size is declared or to the declared qualifier array size.

1451 NOTE: The `common.xsd` file defines a string array of qualifier values, `qualifierSArray`. This definition complies
 1452 with the first mapping rule in this clause. Therefore, in the majority of cases, it is not necessary to create a new array
 1453 complex type definition to define a multi-valued qualifier. Rather, it is sufficient to use the `qualifierSArray` type
 1454 defined in `common.xsd`.

1455 The format for a schema for defining a multi-valued qualifier is as follows:

```
1456 <xs:element name="' cim:qualifier-name '" type="' qualifier-array-type '"/>'
```

1457 Where:

- 1458 – `cim:qualifier-name` is the name of the qualifier, qualified by its namespace prefix.
- 1459 – `qualifier-array-type` is typically the `qualifierSArray` type defined in the `common.xsd` file
 1460 (but may be a user-defined type that complies with the mapping rules).

1461 EXAMPLE 1: A generalized example of the mapping of a multi-valued qualifier is as follows:

```
1462 <ns:QualifierName cim:qualifier="true">
1463   value
1464 </ns:QualifierName>
1465 ... // repeat QualifierName elements for each member of the array
```

1466 Where:

- 1467 – `QualifierName` is the name of the qualifier.
- 1468 – `ns` is the namespace prefix referencing the namespace in which this qualifier is defined.
- 1469 – `n` is a sequential integer that represents the position of the entry in the array.
- 1470 – `value` is the string representation of the qualifier value.

1471 EXAMPLE 2: For example, the mapping of a `ValueMap` qualifier containing three entries ("OK", "Error",
 1472 "Unknown") is as follows:

```
1473 <cimQ:ValueMap cim:qualifier="true">OK</cimQ:ValueMap>
1474 <cimQ:ValueMap cim:qualifier="true">Error</cimQ:ValueMap>
1475 <cimQ:ValueMap cim:qualifier="true">Unknown</cimQ:ValueMap>
```

1476 A complete mapping of all qualifiers is provided in qualifiers.xsd (Annex A.2).

1477 **11.2 Mapping CIM Qualifiers to XSD Elements**

1478 All qualifiers are mapped using the normative rules in 11.1. The qualifiers listed in Table 8 are also
 1479 mapped directly into XSD features.

1480 **Table 8 – CIM Qualifiers Mapped to XSD Elements**

CIM Qualifier	MOF Example	Mapped to XSD Structure
Embedded Instance	[EmbeddedInstance ("Class")]	xs:anyType (normatively defined in 9.2.5.1)
Embedded Object	[EmbeddedObject]	xs:anyType (normatively defined in 9.2.5.1)
Key	[Key]	nillable="false" (normatively defined in 9.2.1.1) NOTE: False is the default value of the nillable attribute and therefore may not be explicitly expressed in the schema.
IN	[IN] / [IN (true)]	The CIM input parameter is mapped to an element in the complex type for the input message GED (normatively defined in 9.5.1).
MaxLen	[MaxLen (1024)]	Mapped to a restriction using xs:maxLength on a string datatype. Required, with the following exception: A qualifier value of NULL shall not be mapped. For example: <pre data-bbox="716 993 1421 1289"><xs:element name="PropName"> <xs:complexType> <xs:simpleContent> <xs:restriction base="cim:cimString"> <xs:maxLength value="1024"/> <xs:anyAttribute .../> </xs:restriction> </xs:simpleContent> </xs:complexType> </xs:element></pre>
MaxValue	[MaxValue (100)]	Mapped to a restriction using xs:maxInclusive on an integer datatype. Required, with the following exception: A qualifier value of NULL, which indicates the largest value allowed by the type, should not be mapped. For example: <pre data-bbox="716 1486 1421 1778"><xs:element name="PropName"> <xs:complexType> <xs:simpleContent> <xs:restriction base="cim:cimUnsignedShort"> <xs:maxInclusive value="100"/> <xs:anyAttribute ... /> </xs:restriction> </xs:simpleContent> </xs:complexType> </xs:element></pre>

CIM Qualifier	MOF Example	Mapped to XSD Structure
MinLen	[MinLen (10)]	<p>Mapped to a restriction using <code>xs:minLength</code> on a string datatype. Required, with the following exception:</p> <p>A qualifier value of 0 should not be mapped.</p> <p>For example:</p> <pre data-bbox="716 409 1421 699"> <xs:element name="PropName"> <xs:complexType> <xs:simpleContent> <xs:restriction base="cim:cimString"> <xs:minLength value="10"/> <xs:anyAttribute ... /> </xs:restriction> </xs:simpleContent> </xs:complexType> </xs:element> </pre>
MinValue	[MinValue (10)]	<p>Mapped to a restriction using <code>xs:minInclusive</code> on an integer datatype. Required, with the following exception:</p> <p>A qualifier value of NULL, which indicates the smallest value allowed by the type, should not be mapped.</p> <p>For example:</p> <pre data-bbox="716 892 1421 1186"> <xs:element name="PropName"> <xs:complexType> <xs:simpleContent> <xs:restriction base="cim:cimUnsignedShort"> <xs:minInclusive value="10"/> <xs:anyAttribute . . ./> </xs:restriction> </xs:simpleContent> </xs:complexType> </xs:element> </pre>
OctetString uint8[] string[]	[OctetString]	<p>Normatively defined in 9.2.4.1</p> <p>cim:cimBase64Binary cim:cimHexBinary array</p>
OUT	[OUT]	<p>The CIM output parameter is mapped to an element in the complex type for the output message GEupp</p>

1481 **11.3 Inheritance of Qualifiers**

1482 In addition to inheritance of properties, references, and methods through class inheritance, qualifier
 1483 values on any CIM elements are inherited. However, qualifiers are subject to special rules of inheritance.
 1484 Qualifier inheritance behavior is defined by the Flavors associated with a particular qualifier declaration.

1485 The rules covering qualifier inheritance are summarized in the third column of Table 9. In Table 9, the
 1486 term "overriding CIM elements in any subclasses" refers to CIM properties, references, and methods that
 1487 override other occurrences of the properties, references, or methods in their superclasses and therefore
 1488 form an inheritance chain. Note that duplicate property, reference, or method names that are *not*
 1489 overridden interrupt the inheritance chain for these CIM elements to their superclasses.

1490 **Table 9 – Rules of Qualifier Inheritance**

FLAVOR	Qualifier Inheritance Behavior (Informative)	Metadata Fragment Mapping Behavior
Restricted	The qualifier value pertains only to the CIM element on which it is defined. It is not inherited by any subclasses or overriding CIM elements in these subclasses. EXAMPLE: Abstract	The metadata fragment mapping for the qualifier applies only to the XSD element mapped from the CIM element that has the qualifier value defined. The metadata fragment shall not be carried to corresponding CIM elements in any subclasses.
ToSubclass: EnableOverride	The qualifier value is inherited by any subclasses or overriding CIM elements in these subclasses. The value of the qualifier may be changed in a subclass. EXAMPLE: MaxLen	The metadata fragment mapping for the qualifier applies to the XSD element mapped from the CIM element that has the qualifier value defined. In addition, the metadata fragment shall be carried to any subclasses or overriding CIM elements in these subclasses. In addition, the metadata fragment shall reflect the qualifier value provided on the corresponding CIM element. Unless overridden on the corresponding CIM element, the metadata fragment shall have the same value as the defined value in the superclass.
ToSubclass: DisableOverride	The qualifier value is inherited by any subclasses or overriding CIM elements in these subclasses. The value of the qualifier must not be changed in a subclass. EXAMPLE: Key	The metadata fragment mapping for the qualifier applies to the XSD element mapped from the CIM element that has the qualifier value defined. In addition, the metadata fragment shall be carried to any subclasses or overriding CIM elements in these subclasses.

1491

1492 **12 Qualifier Annotations**

1493 This clause defines structures which may be included in XML schema files as annotations to describe the
 1494 qualifiers of CIM classes and properties that are included in the CIM schema declarations. The structures
 1495 defined here are different from the structures used for the metadata fragments specified in clause 11 and
 1496 Annex B.4. It is not intended that there be compatibility between the two treatments of qualifier
 1497 information.

1498 A schema file for a CIM class may include annotations to describe all the qualifiers present in the CIM
 1499 schema declaration that defines the class. Qualifiers modifying class and property declarations both are
 1500 included. Such qualifier descriptions may be included in XSD files for classes in the CIM schema or in an
 1501 extension schema.

1502 The elements describing qualifiers are structured in the <xs:appinfo> element of the <xs:annotation>
 1503 element for a class or property. Two new XML elements are used in the annotations to describe all the
 1504 qualifiers and the location of the class in the CIM hierarchy: Qualifier and Class. These elements are
 1505 defined in the WS-CIM XML namespace.

1506 Because the qualifier information is contained in annotations, which do not have normative impact on the
 1507 schema, the schema file can still be used to validate class instances. That is, inclusion of qualifier
 1508 information is upward compatible from previous versions not containing such information. However,
 1509 because of the additional size of the qualifier annotations, it may be advisable to maintain two versions of
 1510 schema files, ones containing qualifier descriptions and smaller ones without annotations.

1511 Examples are included in-line to help illustrate application of the rules. Most examples are derived from
 1512 the class CIM_ComputerSystem.

- 1513 • If an XML schema file includes qualifier information as described in this clause, then all the
 1514 normative rules in this clause shall apply.
- 1515 • An XML schema shall include any qualifier that is declared in the schema for a class,
 1516 property, or reference; and any qualifier that is inherited by the class from a superclass.
- 1517 • If a CIM class definition contains any qualifier information, then the XML schema element
 1518 that defines the class name shall include an <xs:annotation> element as defined in this
 1519 clause.

1520

```
1521 <xs:element name="CIM_ComputerSystem" type="class:CIM_ComputerSystem_Type">
1522   <xs:annotation>
1523     <xs:appinfo . . . >
1524       . . .
1525   </xs:appinfo>
1526 </xs:annotation>
1527 </xs:element>
```

1528

- 1529 • If a CIM property or reference definition contains any qualifier information, then the XML
 1530 schema element that defines the GED of the property or reference shall include an
 1531 <xs:annotation> element as defined in this clause.

1532

```
1533 <xs:element name="NameFormat" nillable="true">
1534   <xs:annotation>
1535     <xs:appinfo . . . >
1536       . . .
1537   </xs:appinfo>
1538 </xs:annotation>
1539 </xs:element>
```

1540

- 1541
- The <xs:annotation> element shall be a direct child of the defining <xs:element>.
- 1542
- The <xs:annotation> element shall include one <xs:appinfo> element containing qualifier information for that property or class.
- 1543
- 1544
- The <xs:appinfo . . . > element shall include the attribute
- 1545
- source="http://schemas.dmtf.org/wbem/wscim/1/common".

1546

1547 `<xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">`

1548

- 1549
- An XML schema shall state one or more <cim:Qualifier name="..."> element for every
- 1550
- qualifier explicitly included in the CIM schema.
- 1551
- The value of the name="..." attribute shall be the name of a qualifier from the CIM schema
- 1552
- declaration of the class or property.

1553 12.1 Extension Qualifiers

1554 Note that qualifiers may be defined by the CIM architecture in [DSP0004](#) or may be defined as extensions

1555 by vendors and users. Extension qualifiers reside in the CIM namespace along with CIM-defined

1556 qualifiers. An extension schema that uses vendor-defined qualifiers uses the same cim:Qualifier element

1557 in the annotation.

- 1558
- An extension schema that uses vendor-defined qualifiers shall use the cim:Qualifier
- 1559
- element with the extension qualifier name in the name="..." attribute.

1560 Example: a vendor defines a qualifier Foo with flavor ToSubclass, This could appear in the

1561 annotation as

1562

1563 `<cim:Qualifier name="Foo" type="cim:cimString">...</cim:Qualifier>`

1564

1565 Extension qualifier names cannot conflict with CIM-defined qualifier names. Client programs may be able

1566 to perform some processing on extension qualifiers for which they do not know the semantics.

1567 12.2 Qualifier Values

1568 The values of qualifiers are declared using CIM datatypes. See clause 8 for the correspondence of XML

1569 and CIM datatypes.

- 1570
- The datatypes of qualifier values in annotations shall be specified using CIM datatypes.

1571

1572 `<cim:Qualifier name="Description" type="cim:cimString">`

1573

- 1574
- The datatype of a qualifier value shall be appropriate to the qualifier value.

1575

1576 `<cim:Qualifier name="Description" type="cim:cimString">`

1577 `<cim:Qualifier name="Maxlen" type="cim:cimUnsignedShort">`

1578 `<cim:Qualifier name="Association" type="cim:cimBoolean">`

1579 `<cim:Qualifier name="Value" type="cim:cimString" . . . >`

1580

- 1581 • The value of a qualifier element shall be the "effective qualifier value" as specified in
1582 [DSP0004](#) of the most derived declaration in the class hierarchy; that is, the value of the
1583 qualifier from the most recent override, or the value of the qualifier from its original
1584 declaration if there are no overrides in the inheritance path.
- 1585 • The value of a qualifier element shall preserve white space that is present in the CIM
1586 schema declaration of the qualifier.
- 1587 • The value of a qualifier element shall be represented in a format appropriate to its
1588 datatype, as defined in clause 8.

1589

```
1590 <xs:element name="CreationClassName">
1591   <xs:annotation>
1592     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
1593       . . .
1594       <cim:Qualifier name="Key" type="cim:cimBoolean">true</cim:Qualifier>
1595       <cim:Qualifier name="Description" type="cim:cimString">
1596         . . .
1597     </cim:Qualifier>
1598     <cim:Qualifier name="MaxLen" type="cim:cimUnsignedInt">256</cim:Qualifier>
1599   </xs:appinfo>
1600 </xs:annotation>
1601   . . .
1602 </xs:element>
```

1603

- 1604 • With the exception of array-typed elements (see relevant rule below), `cim:Qualifier`
1605 elements may appear in any order inside the `xs:appinfo` group.

1606 Qualifiers such as `Value` and `ValueMap` are arrays that may contain multiple elements. In the CIM
1607 schema syntax, these are parallel (indexed) arrays of comma-separated strings. In XML these are
1608 represented by multiple instances of an element.

- 1609 • A schema shall represent an array-typed CIM qualifier as a sequence of `<cim:Qualifier`
1610 `name="...">` elements.
- 1611 • The `cim:Qualifier` elements representing array elements shall appear as a contiguous
1612 group in the order given in the CIM schema definition.

1613

```
1614 In CIM schema:
1615   ModelCorrespondence{ "CIM_ManagedSystemElement.DetailedStatus",
1616                       "CIM_ManagedSystemElement.HealthState" }
1617 in XML schema:
1618   <cim:Qualifier name="Modelcorrespondence"
1619     type="cim:cimString">CIM_ManagedSystemElement.DetailedStatus</cim:Qualifier>
1620   <cim:Qualifier name="Modelcorrespondence"
```

1621 `type="cim:cimString">CIM_ManagedSystemElement.HealthState</cim:Qualifier>`

1622

1623 • The Values and ValueMap parallel arrays from a CIM property definition shall be combined
1624 in the XML schema into a single list.

1625 • If the property has a ValueMap qualifier, then each entry of the ValueMap qualifier array
1626 shall be represented by a `<cim:Qualifier name="Values" ...>` element. The XML element
1627 shall have the value of the Values array entry, if there is one, and the `valuemap="..."`
1628 attribute of the element shall have the value of the corresponding ValueMap entry. If there
1629 is no corresponding Values entry, then the element shall include the `xsi:nil="true"` attribute.

1630 • If the property has no ValueMap qualifier but does have a Values qualifier, then each entry
1631 of the Values qualifier array shall be represented by a `<cim:Qualifier name="Values" ...>`
1632 element. The XML element shall have the value of the Values array entry, and the
1633 `valuemap="..."` attribute shall be absent.

1634 Example:

1635

```
1636 <xs:element name="ResetCapability" nillable="true">
1637   <xs:annotation>
1638     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
1639       . . .
1640       <cim:Qualifier name="Values" type="cim:cimString" valuemap="1">Other</cim:Qualifier>
1641       <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Unknown</cim:Qualifier>
1642       <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Disabled</cim:Qualifier>
1643       <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Enabled</cim:Qualifier>
1644       <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">Not
1645 Implemented</cim:Qualifier>
1646       <cim:Qualifier name="Mappingstrings" type="cim:cimString">MIF.DMTF|System Hardware
1647 Security|001.4</cim:Qualifier>
1648     </xs:appinfo>
1649   </xs:annotation>
1650   . . .
1651 </xs:element>
```

1652

1653 Example:

1654

```
1655 <xs:element name="NameFormat" nillable="true">
1656   <xs:annotation>
1657     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
1658       . . .
1659       <cim:Qualifier name="Values" type="cim:cimString" valuemap="IP" xsi:nil="true"/>
1660       <cim:Qualifier name="Values" type="cim:cimString" valuemap="Other" xsi:nil="true"/>
1661       <cim:Qualifier name="Values" type="cim:cimString" valuemap="Dial" xsi:nil="true"/>
1662       . . .
1663     </xs:appinfo>
```

```

1664     </xs:annotation>
1665     . . .
1666 </xs:element>

```

1667

1668 In CIM all Boolean qualifiers have a value, true or false, though the value may not be stated explicitly in
 1669 some syntaxes. The qualifier description in the schema annotation explicitly includes the truth value.

- 1670 • A schema shall include the value of a boolean qualifier. The value of a boolean qualifier
 1671 shall be "true" or "false" in lower case (following the XML convention).

1672

```

1673 <cim:Qualifier name="Key" type="cim:cimBoolean">true</cim:Qualifier>

```

1674

1675 12.3 Additional Information Not Directly From Qualifiers

1676 Some additional information that is not represented by qualifiers in the schema definition of the class is
 1677 included to specify the inheritance of the class and its properties within the CIM schema hierarchy.

- 1678 • A schema declaration of the class shall include a <cim:Class name="..." superclass="...">
 1679 element in the xs:appinfo element associated with the class being defined.
- 1680 • The cim:Class name="..." attribute value shall specify the full class name.
- 1681 • The cim:Class superclass="..." attribute value shall specify the full class name of the class
 1682 from which this class inherits directly. If the class is a base class, that is, does not inherit
 1683 from any other class, then the superclass="..." attribute shall be absent.

1684

```

1685 <cim:Class name="CIM_ComputerSystem" superclass="CIM_System" />

```

1686

```

1687 <cim:Class name="CIM_ManagedElement" />

```

1688

- 1689 • The schema declaration of a property shall include a <cim:Classorigin name="...">
 1690 element.
- 1691 • The value of the name="..." attribute of the Classorigin element shall be the name of the
 1692 class from which the property is inherited most closely. That is, the classorigin value shall
 1693 be the name of the most derived class in which the property declaration was overridden, or
 1694 the name of the class in which the property was initially declared if there are no overrides
 1695 in the inheritance path.

1696 If a property is defined in a parent class, or overridden most recently in, a parent class, but is not defined
 1697 or overridden in the current class, then that parent class is considered the origin of the property.

1698

```

1699 <xs:element name="Name">
1700   <xs:annotation>
1701     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
1702       <cim:Classorigin name="CIM_System" />

```

1703

```

1704     </xs:appinfo>
1705     </xs:annotation>
1706 </xs:element>
1707

```

1708

1709 If a property is initially defined in the current class, or is overridden in the current class, then the current
 1710 class is the origin.

1711

```

1712 <xs:element name="NameFormat">
1713   <xs:annotation>
1714     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
1715       <cim:Classorigin name="CIM_ComputerSystem"/>
1716       . . .
1717     </xs:appinfo>
1718   </xs:annotation>
1719 </xs:element>

```

1720

1721 If a property declares a default value in the schema, an additional element is used to represent that value.

- 1722 • If a property includes a default value, then the schema declaration of the property shall
 1723 include a <cim:Defaultvalue . . . > element in the </xs:appinfo> element.
- 1724 • The type of the <cim:Defaultvalue . . . > element shall be the CIM datatype of the property.

1725

```

1726 <cim:Defaultvalue type="cim:cimUnsignedShort">5</cim:Defaultvalue>

```

1727

1728 A complete example of the schema for class CIM_ComputerSystem, including qualifier annotations, is
 1729 included in ANNEX D.

1730

1731

ANNEX A (Informative)

Schemas

1732
1733
1734
1735

1736 This annex provides examples of the WS-CIM Schema ([DSP8004](#)), the Qualifiers Schema ([DSP8005](#)),
1737 and the Class Hierarchy Type Schema ([DSP8006](#)).

1738 A.1 Common WS-CIM Schema: DSP8004

1739 This schema contains common definitions.

```

1740 <?xml version="1.0" encoding="utf-8" ?>
1741 <xs:schema targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/common"
1742   xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
1743   xmlns:xs="http://www.w3.org/2001/XMLSchema"
1744   elementFormDefault="qualified">
1745
1746 <!-- The following are runtime attribute definitions -->
1747   <xs:attribute name="Key" type="xs:boolean"/>
1748
1749   <xs:attribute name="Version" type="xs:string"/>
1750
1751 <!-- The following section defines the extended WS-CIM datatypes -->
1752
1753   <xs:complexType name="cimDateTime">
1754     <xs:choice>
1755       <xs:element name="CIM_DateTime" type="xs:string" nillable="true"/>
1756       <xs:element name="Interval" type="xs:duration"/>
1757       <xs:element name="Date" type="xs:date" />
1758       <xs:element name="Time" type="xs:time" />
1759       <xs:element name="Datetime" type="xs:dateTime"/>
1760     </xs:choice>
1761     <xs:anyAttribute namespace="##any" processContents="lax"/>
1762   </xs:complexType>
1763
1764   <xs:complexType name="cimUnsignedByte">
1765     <xs:simpleContent>
1766       <xs:extension base="xs:unsignedByte">
1767         <xs:anyAttribute namespace="##any" processContents="lax"/>
1768       </xs:extension>
1769     </xs:simpleContent>
1770   </xs:complexType>
1771
1772   <xs:complexType name="cimByte">
1773     <xs:simpleContent>
1774       <xs:extension base="xs:byte">
1775         <xs:anyAttribute namespace="##any" processContents="lax"/>
1776       </xs:extension>
1777     </xs:simpleContent>
1778   </xs:complexType>
1779
1780   <xs:complexType name="cimUnsignedShort">
1781     <xs:simpleContent>
1782       <xs:extension base="xs:unsignedShort">

```

```
1783     <xs:anyAttribute namespace="##any" processContents="lax"/>
1784   </xs:extension>
1785 </xs:simpleContent>
1786 </xs:complexType>
1787
1788 <xs:complexType name="cimShort">
1789   <xs:simpleContent>
1790     <xs:extension base="xs:short">
1791       <xs:anyAttribute namespace="##any" processContents="lax"/>
1792     </xs:extension>
1793   </xs:simpleContent>
1794 </xs:complexType>
1795
1796 <xs:complexType name="cimUnsignedInt">
1797   <xs:simpleContent>
1798     <xs:extension base="xs:unsignedInt">
1799       <xs:anyAttribute namespace="##any" processContents="lax"/>
1800     </xs:extension>
1801   </xs:simpleContent>
1802 </xs:complexType>
1803
1804 <xs:complexType name="cimInt">
1805   <xs:simpleContent>
1806     <xs:extension base="xs:int">
1807       <xs:anyAttribute namespace="##any" processContents="lax"/>
1808     </xs:extension>
1809   </xs:simpleContent>
1810 </xs:complexType>
1811
1812 <xs:complexType name="cimUnsignedLong">
1813   <xs:simpleContent>
1814     <xs:extension base="xs:unsignedLong">
1815       <xs:anyAttribute namespace="##any" processContents="lax"/>
1816     </xs:extension>
1817   </xs:simpleContent>
1818 </xs:complexType>
1819
1820 <xs:complexType name="cimLong">
1821   <xs:simpleContent>
1822     <xs:extension base="xs:long">
1823       <xs:anyAttribute namespace="##any" processContents="lax"/>
1824     </xs:extension>
1825   </xs:simpleContent>
1826 </xs:complexType>
1827
1828 <xs:complexType name="cimString">
1829   <xs:simpleContent>
1830     <xs:extension base="xs:string">
1831       <xs:anyAttribute namespace="##any" processContents="lax"/>
1832     </xs:extension>
1833   </xs:simpleContent>
1834 </xs:complexType>
1835
1836 <xs:complexType name="cimBoolean">
1837   <xs:simpleContent>
1838     <xs:extension base="xs:boolean">
1839       <xs:anyAttribute namespace="##any" processContents="lax"/>
1840     </xs:extension>
```

```

1841     </xs:simpleContent>
1842 </xs:complexType>
1843
1844 <xs:complexType name="cimFloat">
1845   <xs:simpleContent>
1846     <xs:extension base="xs:float">
1847       <xs:anyAttribute namespace="##any" processContents="lax"/>
1848     </xs:extension>
1849   </xs:simpleContent>
1850 </xs:complexType>
1851
1852 <xs:complexType name="cimDouble">
1853   <xs:simpleContent>
1854     <xs:extension base="xs:double">
1855       <xs:anyAttribute namespace="##any" processContents="lax"/>
1856     </xs:extension>
1857   </xs:simpleContent>
1858 </xs:complexType>
1859
1860 <xs:complexType name="cimChar16">
1861   <xs:simpleContent>
1862     <xs:restriction base="cim:cimString">
1863       <xs:maxLength value="1"/>
1864       <xs:anyAttribute namespace="##any" processContents="lax"/>
1865     </xs:restriction>
1866   </xs:simpleContent>
1867 </xs:complexType>
1868
1869 <xs:complexType name="cimBase64Binary">
1870   <xs:simpleContent>
1871     <xs:extension base="xs:base64Binary">
1872       <xs:anyAttribute namespace="##any" processContents="lax"/>
1873     </xs:extension>
1874   </xs:simpleContent>
1875 </xs:complexType>
1876
1877 <xs:complexType name="cimHexBinary">
1878   <xs:simpleContent>
1879     <xs:extension base="xs:hexBinary">
1880       <xs:anyAttribute namespace="##any" processContents="lax"/>
1881     </xs:extension>
1882   </xs:simpleContent>
1883 </xs:complexType>
1884
1885 <xs:complexType name="cimAnySimpleType">
1886   <xs:simpleContent>
1887     <xs:extension base="xs:anySimpleType">
1888       <xs:anyAttribute namespace="##any" processContents="lax"/>
1889     </xs:extension>
1890   </xs:simpleContent>
1891 </xs:complexType>
1892
1893 <xs:complexType name="cimReference">
1894   <xs:sequence>
1895     <xs:any namespace="##other" maxOccurs="unbounded" processContents="lax"/>
1896   </xs:sequence>
1897   <xs:anyAttribute namespace="##any" processContents="lax"/>
1898 </xs:complexType>

```

```

1899
1900 <!-- The following datatypes are used exclusively to define metadata fragments -->
1901 <xs:attribute name="qualifier" type="xs:boolean"/>
1902
1903 <xs:complexType name="qualifierString">
1904 <xs:simpleContent>
1905 <xs:extension base="cim:cimString">
1906 <xs:attribute ref="cim:qualifier" use="required"/>
1907 </xs:extension>
1908 </xs:simpleContent>
1909 </xs:complexType>
1910
1911 <xs:complexType name="qualifierBoolean">
1912 <xs:simpleContent>
1913 <xs:extension base="cim:cimBoolean">
1914 <xs:attribute ref="cim:qualifier" use="required"/>
1915 </xs:extension>
1916 </xs:simpleContent>
1917 </xs:complexType>
1918
1919 <xs:complexType name="qualifierUInt32">
1920 <xs:simpleContent>
1921 <xs:extension base="cim:cimUnsignedInt">
1922 <xs:attribute ref="cim:qualifier" use="required"/>
1923 </xs:extension>
1924 </xs:simpleContent>
1925 </xs:complexType>
1926
1927 <xs:complexType name="qualifierSInt64">
1928 <xs:simpleContent>
1929 <xs:extension base="cim:cimLong">
1930 <xs:attribute ref="cim:qualifier" use="required"/>
1931 </xs:extension>
1932 </xs:simpleContent>
1933 </xs:complexType>
1934
1935 <xs:complexType name="qualifierSArray">
1936 <xs:complexContent>
1937 <xs:extension base="cim:qualifierString"/>
1938 </xs:complexContent>
1939 </xs:complexType>
1940
1941 <!-- The following element is to be used only for defining metadata -->
1942 <xs:element name="DefaultValue" type="xs:anySimpleType" />
1943
1944 </xs:schema>

```

1945 A.2 Qualifiers Schema: DSP8005

1946 The following schema is an example of the qualifiers schema that is based on CIM Schema 2.13.1.
 1947 Future versions of CIM Schema may add or delete qualifiers, which would be reflected in the
 1948 corresponding qualifiers.xsd file.

```

1949 <?xml version="1.0" encoding="utf-8" ?>
1950 <xs:schema
1951 targetNamespace="http://schemas.dmtf.org/wbem/ws-cim/1/cim-schema/2/qualifiers"
1952 xmlns:cimQ="http://schemas.dmtf.org/wbem/ws-cim/1/cim-schema/2/qualifiers"
1953 xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"

```

```

1954     xmlns:xs="http://www.w3.org/2001/XMLSchema"
1955     elementFormDefault="qualified">
1956
1957     <xs:import
1958         namespace="http://schemas.dmtf.org/wbem/wscim/1/common"
1959         schemaLocation="http://schemas.dmtf.org/wbem/wscim/1/common.xsd"/>
1960
1961     <xs:element name="Abstract" type="cim:qualifierBoolean"/>
1962     <xs:element name="Aggregate" type="cim:qualifierBoolean"/>
1963     <xs:element name="Aggregation" type="cim:qualifierBoolean"/>
1964     <xs:element name="ArrayType" type="cim:qualifierString"/>
1965     <xs:element name="Association" type="cim:qualifierBoolean"/>
1966     <xs:element name="BitMap" type="cim:qualifierSArray"/>
1967     <xs:element name="BitValues" type="cim:qualifierSArray"/>
1968     <xs:element name="ClassConstraint" type="cim:qualifierSArray"/>
1969     <xs:element name="Counter" type="cim:qualifierBoolean"/>
1970     <xs:element name="Composition" type="cim:qualifierBoolean"/>
1971     <xs:element name="Deprecated" type="cim:qualifierSArray"/>
1972     <xs:element name="Description" type="cim:qualifierString"/>
1973     <xs:element name="DisplayName" type="cim:qualifierString"/>
1974     <xs:element name="DN" type="cim:qualifierBoolean"/>
1975     <xs:element name="EmbeddedInstance" type="cim:qualifierBoolean"/>
1976     <xs:element name="EmbeddedObject" type="cim:qualifierBoolean"/>
1977     <xs:element name="Exception" type="cim:qualifierBoolean"/>
1978     <xs:element name="Experimental" type="cim:qualifierBoolean"/>
1979     <xs:element name="Gauge" type="cim:qualifierBoolean"/>
1980     <xs:element name="In" type="cim:qualifierBoolean"/>
1981     <xs:element name="Indication" type="cim:qualifierBoolean"/>
1982     <xs:element name="Key" type="cim:qualifierBoolean"/>
1983     <xs:element name="MappingStrings" type="cim:qualifierSArray"/>
1984     <xs:element name="Max" type="cim:qualifierUInt32"/>
1985     <xs:element name="MethodConstraint" type="cim:qualifierSArray"/>
1986     <xs:element name="Min" type="cim:qualifierUInt32"/>
1987     <xs:element name="MaxLen" type="cim:qualifierUInt32"/>
1988     <xs:element name="MaxValue" type="cim:qualifierSInt64"/>
1989     <xs:element name="MinLen" type="cim:qualifierUInt32"/>
1990     <xs:element name="MinValue" type="cim:qualifierSInt64"/>
1991     <xs:element name="Revision" type="cim:qualifierString"/>           <!-- Is Deprecated -->
1992     <xs:element name="ModelCorrespondence" type="cim:qualifierSArray"/>
1993     <xs:element name="NullValue" type="cim:qualifierString"/>
1994     <xs:element name="OctetString" type="cim:qualifierBoolean"/>
1995     <xs:element name="Out" type="cim:qualifierBoolean"/>
1996     <xs:element name="Override" type="cim:qualifierString"/>
1997     <xs:element name="Propagated" type="cim:qualifierString"/>
1998     <xs:element name="PropertyConstraint" type="cim:qualifierSArray"/>
1999     <xs:element name="Read" type="cim:qualifierBoolean"/>
2000     <xs:element name="Required" type="cim:qualifierBoolean"/>
2001     <xs:element name="Schema" type="cim:qualifierString"/>
2002     <xs:element name="Static" type="cim:qualifierBoolean"/>
2003     <xs:element name="Terminal" type="cim:qualifierBoolean"/>
2004     <xs:element name="Units" type="cim:qualifierString"/>
2005     <xs:element name="UMLPackagePath" type="cim:qualifierString"/>
2006     <xs:element name="ValueMap" type="cim:qualifierSArray"/>

```

```

2007 <xs:element name="Values" type="cim:qualifierSArray"/>
2008 <xs:element name="Version" type="cim:qualifierString"/>
2009 <xs:element name="Weak" type="cim:qualifierBoolean"/>
2010 <xs:element name="Write" type="cim:qualifierBoolean"/>
2011
2012 <!-- Qualifier defined by DMTF for a future release of CIM Schema -->
2013 <!-- Included in this version at the request of the WSDM-CIM mapping team -->
2014 <xs:element name="Correlatable" type="cim:qualifierSArray"/>
2015
2016 <!-- Following qualifiers are considered to be "Optional Qualifiers" in CIM. -->
2017 <xs:element name="Alias" type="cim:qualifierString"/>
2018 <xs:element name="Delete" type="cim:qualifierBoolean"/>
2019 <xs:element name="Expensive" type="cim:qualifierBoolean"/>
2020 <xs:element name="IfDeleted" type="cim:qualifierBoolean"/>
2021 <xs:element name="Invisible" type="cim:qualifierBoolean"/>
2022 <xs:element name="Large" type="cim:qualifierBoolean"/>
2023 <xs:element name="Provider" type="cim:qualifierString"/>
2024 <xs:element name="PropertyUsage" type="cim:qualifierString"/>
2025 <xs:element name="Syntax" type="cim:qualifierString"/>
2026 <xs:element name="SyntaxType" type="cim:qualifierString"/>
2027 <xs:element name="TriggerType" type="cim:qualifierString"/>
2028 <xs:element name="UnknownValues" type="cim:qualifierSArray"/>
2029 <xs:element name="UnsupportedValues" type="cim:qualifierSArray"/>
2030
2031 </xs:schema>

```

2032 A.3 Class Hierarchy Type Schema: DSP8006

2033 The complex type definition in the following schema provides the type of GEDs that describe the CIM
 2034 Schema subclass / superclass hierarchy. The element `ClassHierarchy` may be used by protocols as
 2035 an element in XML instance documents of a CIM instance that contains a value representing the subclass
 2036 / superclass hierarchy of a class. Its presence as an element in an instance document would be covered
 2037 by the `xs:any` in the WS-CIM schema of the instance's class.

```

2038 <?xml version="1.0" encoding="utf-8"?>
2039 <xs:schema
2040   targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/classhiertype"
2041   xmlns:ctype="http://schemas.dmtf.org/wbem/wscim/1/classhiertype"
2042   xmlns:xs="http://www.w3.org/2001/XMLSchema">
2043   <xs:complexType name="ClassHierarchyType">
2044     <xs:sequence>
2045       <xs:any minOccurs="0" namespace="##any" processContents="lax" />
2046     </xs:sequence>
2047   </xs:complexType>
2048
2049   <xs:element name="ClassHierarchy" type="ctype:ClassHierarchyType"/>
2050
2051 </xs:schema>

```

ANNEX B (Informative)

Examples

2056 This annex contains examples of converting MOF definitions of several classes into XML Schema, WSDL
2057 fragments, and metadata fragments. Although the classes are fictional creations used to illustrate
2058 different features of the conversion, the classes are based on actual CIM classes.

2059 B.1 MOF Definitions

2060 This clause contains the MOF definitions that are converted in these examples.

2061 B.1.1 EX_BaseComponent

```

2062 [Abstract, Version ( "2.x" ), Description (
2063     "EX_BaseComponent serves as an example base CIM class.")]
2064 class EX_BaseComponent {
2065     [Description (
2066         "A datetime value indicating when the object was installed.")]
2067     datetime InstallDate;
2068     [Description (
2069         "The Name property defines the label by which the object is "
2070         "known."),
2071     MaxLen ( 1024 ), Required]
2072     string Name;
2073     [Description (
2074         "A set of descriptive statements that can be used to describe the "
2075         "state of a Component."),
2076     ArrayType ( "Indexed" ) ]
2077     string StatusDescriptions[];
2078     [Description (
2079         "A descriptive code representing operational health of a Component."),
2080     ValueMap { "OK", "Error", "Unknown" }, MaxLen ( 10 ) ]
2081     string HealthStatus;
2082 };

```

2083 B.1.2 EX_DerivedComponent

```

2084 [Version ( "2.x" ), Description (
2085     "This class extends EX_BaseComponent.")]
2086 class EX_DerivedComponent : EX_BaseComponent {
2087     [Description (
2088         "EnabledState is an integer enumeration that indicates the "
2089         "enabled and disabled states of a derived Component."),
2090     ValueMap { "0", "1", "2", "3" },
2091     Values { "Unknown", "Other", "Enabled", "Disabled" } ]
2092     uint16 EnabledState=3;
2093     [Description (
2094         "Boolean flag indicating availability of a Component.") ]
2095     boolean AvailableFlag;
2096     [Description (
2097         "Requests that the state of the element be changed to the "
2098         "value specified in the RequestedState parameter."),
2099     ValueMap { "0", "1", "2", "3..32767", "32768..65535" },
2100     Values { "Completed with No Error", "Not Supported",

```

```

2101         "Failed", "DMTF Reserved", "Vendor Specific" } ]
2102     uint32 RequestStateChange(
2103         [IN, Description (
2104             "The state requested for the Component."),
2105             ValueMap { "2", "3", "4" },
2106             Values { "Enabled", "Disabled" "Shutdown" } ]
2107     uint16 RequestedState,
2108     [IN ( false ), OUT, Description (
2109         "Reference to an instance of some class (undefined in this "
2110         "example) that is returned upon completion of the operation.")]
2111     CIM_SomeClass REF ResultClass,
2112     [IN, Description (
2113         "A timeout period that specifies the maximum amount of "
2114         "time that the client expects the transition to the new "
2115         "state to take. ")]
2116     datetime TimeoutPeriod);
2117 };

```

2118 B.1.3 EX_AssociationComponent

```

2119     [Association, Version ( "2.x" ), Description (
2120         "Indicates that two entites are associated.")]
2121 class EX_Association {
2122     [Key, Description (
2123         "AssociatingComponent represents one Component is "
2124         "associated with the Component referenced as AssociatedComponent.")]
2125     EX_BaseComponent REF AssociatingComponent;
2126     [Key, Description (
2127         "AssociatedComponent represents another Component (up to 4) that "
2128         "is associated with the Component referenced as "
2129         "AssociatingComponent."),
2130         Max ( 4 )]
2131     EX_BaseComponent REF AssociatedComponent;
2132     [Description (
2133         "The point in time that the Components were associated.")]
2134     datetime WhenAssociated;
2135     [Description (
2136         "Boolean indicating whether the association is maintained.")]
2137     boolean AssocMaintained;
2138 };

```

2139 B.2 XSD

2140 This clause shows the XML Schema files that would result from the application of this specification to the
 2141 preceding example CIM classes.

2142 B.2.1 EX_BaseComponent

```

2143 <?xml version="1.0" encoding="utf-8"?>
2144 <xs:schema
2145     targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/EX_BaseComponent"
2146     xmlns:class="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/EX_BaseComponent"
2147     xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
2148     xmlns:xs="http://www.w3.org/2001/XMLSchema"
2149     ...>
2150 <xs:import
2151     namespace="http://schemas.dmtf.org/wbem/wscim/1/common"
2152     schemaLocation="http://schemas.dmtf.org/wbem/wscim/1/common.xsd"/>
2153 <xs:element name="InstallDate" type="cim:cimDateTime" nillable="true"/>

```



```

2154 <xs:element name="Name">
2155   <xs:complexType>
2156     <xs:simpleContent>
2157       <xs:restriction base="cim:cimString">
2158         <xs:maxLength value="1024"/>
2159         <xs:anyAttribute namespace="##any" processContents="lax"/>
2160       </xs:restriction>
2161     </xs:simpleContent>
2162   </xs:complexType>
2163 </xs:element>
2164 <xs:element name="StatusDescriptions" type="cim:cimString"/>
2165 <xs:element name="HealthStatus" nillable="true">
2166   <xs:complexType>
2167     <xs:simpleContent>
2168       <xs:restriction base="cim:cimString">.
2169         <xs:enumeration value="OK"/>
2170         <xs:enumeration value="Error"/>
2171         <xs:enumeration value="Unknown"/>
2172         <xs:maxLength value="10"/>
2173         <xs:anyAttribute namespace="##any" processContents="lax"/>
2174       </xs:restriction>
2175     </xs:simpleContent>
2176   </xs:complexType>
2177 </xs:element>
2178 <xs:complexType name="EX_BaseComponent_Type">
2179   <xs:sequence>
2180     <xs:element ref="class:HealthStatus" minOccurs="0"/>
2181     <xs:element ref="class:InstallDate" minOccurs="0"/>
2182     <xs:element ref="class:Name"/>
2183     <xs:element ref="class:StatusDescriptions" minOccurs="0" maxOccurs="unbounded"/>
2184     <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
2185   </xs:sequence>
2186   <xs:anyAttribute namespace="##any" processContent="lax"/>
2187 </xs:complexType>
2188 <xs:element name="EX_BaseComponent" type="class:EX_BaseComponent_Type"/>
2189 </xs:schema>

```

2190 B.2.2 EX_DerivedComponent

```

2191 <?xml version="1.0" encoding="utf-8"?>
2192 <xs:schema
2193   targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/EX_DerivedComponent"
2194   xmlns:class="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/EX_DerivedComponent"
2195   xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
2196   xmlns:xs="http://www.w3.org/2001/XMLSchema"
2197   ...>
2198 <xs:import
2199   namespace="http://schemas.dmtf.org/wbem/wscim/1/common"
2200   schemaLocation="http://schemas.dmtf.org/wbem/wscim/1/common.xsd"/>
2201 <xs:element name="InstallDate" type="cim:cimDateTime" nillable="true"/>
2202 <xs:element name="Name">
2203   <xs:complexType>
2204     <xs:simpleContent>
2205       <xs:restriction base="cim:cimString">
2206         <xs:maxLength value="1024"/>
2207         <xs:anyAttribute namespace="##any" processContents="lax"/>
2208       </xs:restriction>
2209     </xs:simpleContent>
2210   <xs:complexType>

```

```

2211 </xs:element>
2212 <xs:element name="StatusDescriptions" type="cim:cimString"/>
2213 <xs:element name="HealthStatus" nillable="true">
2214 <xs:complexType>
2215 <xs:simpleContent>
2216 <xs:restriction base="cim:cimString">
2217 <xs:enumeration value="OK"/>
2218 <xs:enumeration value="Error"/>
2219 <xs:enumeration value="Unknown"/>
2220 <xs:maxLength value="10"/>
2221 <xs:anyAttribute namespace="##any" processContents="lax"/>
2222 </xs:restriction>
2223 </xs:simpleContent>
2224 </xs:complexType>
2225 </xs:element>
2226 <xs:element name="EnabledState" nillable="true">
2227 <xs:complexType>
2228 <xs:simpleContent>
2229 <xs:restriction base="cim:cimUnsignedShort">
2230 <xs:enumeration value="0"/>
2231 <xs:enumeration value="1"/>
2232 <xs:enumeration value="2"/>
2233 <xs:enumeration value="3"/>
2234 <xs:anyAttribute namespace="##any" processContents="lax"/>
2235 </xs:restriction>
2236 </xs:simpleContent>
2237 </xs:complexType>
2238 </xs:element>
2239 <xs:element name="AvailableFlag" type="cim:cimBoolean" nillable="true"/>
2240 <xs:complexType name="EX_DerivedComponent_Type">
2241 <xs:sequence>
2242 <xs:element ref="class:AvailableFlag" minOccurs="0"/>
2243 <xs:element ref="class:EnabledState" minOccurs="0"/>
2244 <xs:element ref="class:HealthStatus" minOccurs="0"/>
2245 <xs:element ref="class:InstallDate" minOccurs="0"/>
2246 <xs:element ref="class:Name"/>
2247 <xs:element ref="class:StatusDescriptions" minOccurs="0" maxOccurs="unbounded"/>
2248 <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
2249 </xs:sequence>
2250 <xs:anyAttribute namespace="##any" processContent="lax"/>
2251 </xs:complexType>
2252 <xs:element name="EX_DerivedComponent" type="class:EX_DerivedComponent_Type"/>
2253 <xs:element name="RequestStateChange_INPUT">
2254 <xs:complexType>
2255 <xs:sequence>
2256 <xs:element name="RequestedState" nillable="true">
2257 <xs:complexType>
2258 <xs:simpleContent>
2259 <xs:restriction base="cim:cimUnsignedShort">
2260 <xs:enumeration value="2"/>
2261 <xs:enumeration value="3"/>
2262 <xs:enumeration value="4">
2263 <xs:anyAttribute namespace="##any" processContents="lax"/>
2264 </xs:restriction>
2265 </xs:simpleContent>
2266 </xs:complexType>
2267 </xs:element>
2268 <xs:element name="TimeoutPeriod" type="cim:cimDateTime" nillable="true"/>

```

```

2269     </xs:sequence>
2270   </xs:complexType>
2271 </xs:element>
2272 <xs:element name="RequestStateChange_OUTPUT">
2273   <xs:complexType>
2274     <xs:sequence>
2275       <xs:element name="ResultClass" type="cim:cimReference" nillable="true"/>
2276       <xs:element name="ReturnValue" nillable="true">
2277         <xs:complexType>
2278           <xs:simpleContent>
2279             <xs:restriction base="cim:cimAnySimpleType">
2280               <xs:simpleType>
2281                 <xs:union>
2282                   <xs:simpleType>
2283                     <xs:restriction base="xs:unsignedInt">
2284                       <xs:enumeration value="0"/>
2285                       <xs:enumeration value="1"/>
2286                       <xs:enumeration value="2"/>
2287                     </xs:restriction>
2288                   </xs:simpleType>
2289                   <xs:simpleType>
2290                     <xs:restriction base="xs:unsignedInt">
2291                       <xs:minInclusive value="3"/>
2292                       <xs:maxInclusive value="32767"/>
2293                     </xs:restriction>
2294                   </xs:simpleType>
2295                   <xs:simpleType>
2296                     <xs:restriction base="xs:unsignedInt">
2297                       <xs:minInclusive value="32768"/>
2298                       <xs:maxInclusive value="65535"/>
2299                     </xs:restriction>
2300                   </xs:simpleType>
2301                 </xs:union>
2302               </xs:simpleType>
2303               <xs:anyAttribute namespace="##any" processContents="lax"/>
2304             </xs:restriction>
2305           </xs:simpleContent>
2306         </xs:complexType>
2307       </xs:element>
2308     </xs:sequence>
2309   </xs:complexType>
2310 </xs:element>
2311 </xs:schema>

```

2312 **B.2.3 EX_AssociationComponent**

```

2313 <?xml version="1.0" encoding="utf-8"?>
2314 <xs:schema
2315   targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/EX_AssociationComponent"
2316   xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
2317   xmlns:class="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/EX_AssociationComponent"
2318   xmlns:xs="http://www.w3.org/2001/XMLSchema"
2319   ...>
2320 <xs:import
2321   namespace="http://schemas.dmtf.org/wbem/wscim/1/common"
2322   schemaLocation="http://schemas.dmtf.org/wbem/wscim/1/common.xsd"/>
2323 <xs:element name="AssociatingComponent" type="cim:cimReference"/>
2324 <xs:element name="AssociatedComponent" type="cim:cimReference"/>
2325 <xs:element name="WhenAssociated" type="cim:cimDateTime" nillable="true"/>

```

```

2326 <xs:element name="AssocMaintained" type="cim:cimBoolean" nillable="true"/>
2327 <xs:complexType name="EX_AssociationComponent_Type">
2328   <xs:sequence>
2329     <xs:element ref="class:AssociatedComponent" />
2330     <xs:element ref="class:AssociatingComponent" />
2331     <xs:element ref="class:AssocMaintained" minOccurs="0"/>
2332     <xs:element ref="class:WhenAssociated" minOccurs="0"/>
2333     <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
2334   </xs:sequence>
2335   <xs:anyAttribute namespace="##any" processContent="lax"/>
2336 </xs:complexType>
2337 <xs:element name="EX_AssociationComponent" type="class:EX_AssociationComponent_Type"/>
2338 </xs:schema>

```

2339 B.2.4 Class Hierarchy Schema

```

2340 <?xml version="1.0" encoding="utf-8"?>
2341 <xs:schema
2342   targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/classhierarchy"
2343   xmlns:chier="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/classhierarchy"
2344   xmlns:ctype="http://schemas.dmtf.org/wbem/wscim/1/classhiertype"
2345   xmlns:xs="http://www.w3.org/2001/XMLSchema">
2346   <xs:import
2347     namespace="http://schemas.dmtf.org/wbem/wscim/1/classhiertype"
2348     schemaLocation="http://schemas.dmtf.org/wbem/wscim/1/classhiertype.xsd"/>
2349   <xs:element name="EX_BaseComponent_Class">
2350     <xs:complexType>
2351       <xs:complexContent>
2352         <xs:restriction base="ctype:ClassHierarchyType" />
2353       </xs:complexContent>
2354     </xs:complexType>
2355   </xs:element>
2356   <xs:element name="EX_DerivedComponent_Class">
2357     <xs:complexType>
2358       <xs:complexContent>
2359         <xs:restriction base="ctype:ClassHierarchyType">
2360           <xs:sequence>
2361             <xs:element ref="chier:EX_BaseComponent_Class" />
2362           </xs:sequence>
2363         </xs:restriction>
2364       </xs:complexContent>
2365     </xs:complexType>
2366   </xs:element>
2367   <xs:element name="EX_AssociationComponent_Class">
2368     <xs:complexType>
2369       <xs:complexContent>
2370         <xs:restriction base="ctype:ClassHierarchyType" />
2371       </xs:complexContent>
2372     </xs:complexType>
2373   </xs:element>
2374 </xs:schema>

```

2375 B.3 WSDL Fragments

2376 This clause contains the WSDL fragments (`wSDL:message`, `wSDL:operation`) that would result from
 2377 the application of this specification to the `EX_DerivedComponent` class. This class specifies only one
 2378 method, `RequestStateChange`.

```

2379 <?xml version="1.0" encoding="utf-8"?>
2380 <wSDL:definitions
2381     xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
2382     targetNamespace="http://. . .wSDL"
2383     xmlns:cimClass="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/EX_DerivedComponent"
2384     xmlns:thisWSDL="http://. . .wSDL"
2385     ...>
2386 <w:import namespace="http://www.w3.org/2005/08/addressing"
2387     location="http://www.w3.org/2005/08/addressing/ws-addr.xsd"/>
2388 <wSDL:types>
2389     ... <!-- Schema of EX_DerivedComponent -->
2390 </wSDL:types>
2391 <wSDL:message name="RequestStateChange_InputMessage">
2392     <wSDL:part name="body"
2393         element="cimClass:RequestStateChange_INPUT"/>
2394 </wSDL:message>
2395 <wSDL:message name="RequestStateChange_OutputMessage">
2396     <wSDL:part name="body"
2397         element="cimClass:RequestStateChange_OUTPUT"/>
2398 </wSDL:message>
2399 <!-- OPERATION: RequestStateChange
2400     <wSDL:operation name="RequestStateChange">
2401         <wSDL:input name="RequestStateChange_InputMessage"
2402             message="thisWSDL:RequestStateChange_InputMessage"
2403             <wSDL:output name="RequestStateChange_OutputMessage"
2404                 message="thisWSDL:RequestStateChange_OutputMessage"
2405             </wSDL:operation>
2406 -->
2407 </wSDL:definitions>
  
```

2408 B.4 MetaData Fragments

2409 Metadata fragments are generated from the qualifiers that are associated with a class, property,
 2410 reference, method, or parameter. XML documents that incorporate these fragments must import the `cim`
 2411 and `cimQ` namespaces.

2412 B.4.1 EX_BaseComponent

2413 B.4.1.1 Class Qualifiers

```

2414 <cimQ:Abstract cim:qualifier="true">true</cimQ:Abstract>
2415 <cimQ:Version cim:qualifier="true">2.x</cimQ:Version>
2416 <cimQ:Description cim:qualifier="true">
2417     EX_BaseComponent serves as an example base CIM class.
2418 </cimQ:Description>
  
```

2419 B.4.1.2 Property Qualifiers

2420 B.4.1.2.1 HealthStatus

```

2421 <cimQ:Description cim:qualifier="true">
2422     A descriptive code of the operational health of a Component.
2423 </cimQ:Description>
2424 <cimQ:ValueMap cim:qualifier="true">OK</cimQ:ValueMap>
  
```

2425 <cimQ:ValueMap cim:qualifier="true">Error</cimQ:ValueMap>

2426 **B.4.1.2.2 InstallDate**

2427 <cimQ:Description cim:qualifier="true">
 2428 EX_BaseComponent serves as an example base CIM class.
 2429 </cimQ:Description>
 2430 <cimQ:ValueMap cim:qualifier="true">Unknown</cimQ:ValueMap>

2431 **B.4.1.2.3 Name**

2432 <cimQ:Description cim:qualifier="true">
 2433 The Name property defines the label by which the object is known.
 2434 </cimQ:Description>
 2435 <cimQ:MaxLen cim:qualifier="true">1024</cimQ:MaxLen>
 2436 <cimQ:Required cim:qualifier="true">true</cimQ:Required>

2437 **B.4.1.2.4 StatusDescriptions**

2438 <cimQ:Description cim:qualifier="true">
 2439 A set of descriptive statements that can be used to describe the state of an Component.
 2440 </cimQ:Description>
 2441 <cimQ:ArrayType cim:qualifier="true">Indexed</cimQ:ArrayType>

2442 **B.4.2 EX_DerivedComponent**

2443 **B.4.2.1 Class Qualifiers**

2444 <cimQ:Version cim:qualifier="true">2.x</cimQ:Version>
 2445 <cimQ:Description cim:qualifier="true">
 2446 This class extends EX_BaseComponent.
 2447 </cimQ:Description>

2448 **B.4.2.2 Property Qualifiers**

2449 **B.4.2.2.1 AvailableFlag**

2450 <cimQ:Description cim:qualifier="true">
 2451 Boolean flag indicating availability of a Component.

2452 **B.4.2.2.2 EnabledState**

2453 <cimQ:Description cim:qualifier="true">
 2454 EnabledState is an integer enumeration that indicates the enabled
 2455 and disabled states of a derived Component.
 2456 </cimQ:Description>
 2457 <cimQ:ValueMap cim:qualifier="true">0</cimQ:ValueMap>
 2458 <cimQ:ValueMap cim:qualifier="true">1</cimQ:ValueMap>
 2459 <cimQ:ValueMap cim:qualifier="true">2</cimQ:ValueMap>
 2460 <cimQ:ValueMap cim:qualifier="true">3</cimQ:ValueMap>
 2461 <cimQ:Values cim:qualifier="true">Unknown</cimQ:Values>
 2462 <cimQ:Values cim:qualifier="true">Other</cimQ:Values>
 2463 <cimQ:Values cim:qualifier="true">Enabled</cimQ:Values>
 2464 <cimQ:Values cim:qualifier="true">Disabled</cimQ:Values>
 2465 <cim:DefaultValue xsi:type="xs:uint16">3</cim:DefaultValue>
 2466 HealthStatus
 2467 <cimQ:Description cim:qualifier="true">
 2468 A descriptive code of the operational health of a Component.
 2469 </cimQ:Description>
 2470 <cimQ:ValueMap cim:qualifier="true">OK</cimQ:ValueMap>
 2471 <cimQ:ValueMap cim:qualifier="true">Error</cimQ:ValueMap>

```
2472 <cimQ:ValueMap cim:qualifier="true">Unknown</cimQ:ValueMap>
2473 </cimQ:Description>
```

2474 **B.4.2.2.3 InstallDate**

```
2475 <cimQ:Description cim:qualifier="true">
2476     EX_BaseComponent serves as an example base CIM class.
2477 </cimQ:Description>
```

2478 **B.4.2.2.4 Name**

```
2479 <cimQ:Description cim:qualifier="true">
2480     The Name property defines the label by which the object is known.
2481 </cimQ:Description>
2482 <cimQ:MaxLen cim:qualifier="true">1024</cimQ:MaxLen>
2483 <cimQ:Required cim:qualifier="true">true</cimQ:Required>
```

2484 **B.4.2.2.5 StatusDescriptions**

```
2485 <cimQ:Description cim:qualifier="true">
2486     A set of descriptive statements that can used to describe the state of an Component.
2487 </cimQ:Description>
2488 <cimQ:ArrayType cim:qualifier="true">Indexed</cimQ:ArrayType>
```

2489 **B.4.2.2.6 AvailableFlag**

```
2490 <cimQ:Description cim:qualifier="true">
2491     Boolean flag indicating availability of a Component.
2492 </cimQ:Description>
```

2493 **B.4.2.3 Method and Parameter Qualifiers**

2494 **B.4.2.3.1 RequestStatusChange Method**

```
2495 <cimQ:Description cim:qualifier="true">
2496     Requests that the state of the element be changed to the value
2497     specified in the RequestedState parameter.
2498 </cimQ:Description>
2499 <cimQ:ValueMap cim:qualifier="true">0</cimQ:ValueMap>
2500 <cimQ:ValueMap cim:qualifier="true">1</cimQ:ValueMap>
2501 <cimQ:ValueMap cim:qualifier="true">..</cimQ:ValueMap>
2502 <cimQ:ValueMap cim:qualifier="true">4096</cimQ:ValueMap>
2503 <cimQ:ValueMap cim:qualifier="true">4100..32767</cimQ:ValueMap>
2504 <cimQ:ValueMap cim:qualifier="true">32768..65535</cimQ:ValueMap>
2505 <cimQ:Values cim:qualifier="true">Completed with No Error</cimQ:Values>
2506 <cimQ:Values cim:qualifier="true">Not Supported</cimQ:Values>
2507 <cimQ:Values cim:qualifier="true">Unknown or Unspecified Error</cimQ:Values>
2508 <cimQ:Values cim:qualifier="true">Failed</cimQ:Values>
2509 <cimQ:Values cim:qualifier="true">DMTF Reserved</cimQ:Values>
2510 <cimQ:Values cim:qualifier="true">Vendor Specific</cimQ:Values>
```

2511 **B.4.2.3.2 RequestedState Parameter**

```
2512 <cimQ:Description cim:qualifier="true">
2513     The state requested for the Component.
2514 </cimQ:Description>
2515 <cimQ:In cim:qualifier="true">true</cimQ:In>
2516 <cimQ:ValueMap cim:qualifier="true">2</cimQ:ValueMap>
2517 <cimQ:ValueMap cim:qualifier="true">3</cimQ:ValueMap>
2518 <cimQ:ValueMap cim:qualifier="true">4</cimQ:ValueMap>
2519 <cimQ:Values cim:qualifier="true">Enabled</cimQ:Values>
```

2520 <cimQ:Values cim:qualifier="true">Disabled</cimQ:Values>

2521 <cimQ:Values cim:qualifier="true">Shutdown</cimQ:Values>

2522 **B.4.2.3.3 ResultClass Parameter**

2523 <cimQ:Description cim:qualifier="true">

2524 Reference to an instance of some class (undefined in this example)
2525 that is returned upon completion of the operation.

2526 </cimQ:Description>

2527 <cimQ:Out cim:qualifier="true">true</cimQ:Out>

2528 <cimQ:In cim:qualifier="true">false</cimQ:In>

2529 **B.4.2.3.4 TimeoutPeriod Parameter**

2530 <cimQ:Description cim:qualifier="true">

2531 A timeout period that specifies the maximum amount of time that the
2532 client expects the transition to the new state to take.

2533 </cimQ:Description>

2534 <cimQ:In cim:qualifier="true">true</cimQ:In>

2535 **B.4.3 EX_AssociationComponent**

2536 **B.4.3.1 Class Qualifiers**

2537 <cimQ:Version cim:qualifier="true">2.x</cimQ:Version>

2538 <cimQ:Description cim:qualifier="true">

2539 Indicates that two entites are associated.

2540 </cimQ:Description>

2541 <cimQ:Association cim:qualifier="true">true</cimQ:Association>

2542 **B.4.3.2 Property Qualifiers**

2543 **B.4.3.2.1 AssociatedComponent**

2544 <cimQ:Key cim:qualifier="true">true</cimQ:Key>

2545 <cimQ:Description cim:qualifier="true">

2546 AssociatedComponent represents another Component (up to 4) that is associated
2547 with the Component referenced as AssociatingComponent.

2548 </cimQ:Description>

2549 <cimQ:Max cim:qualifier="true">4</cimQ:Max>

2550 **B.4.3.2.2 AssociatingComponent**

2551 <cimQ:Key cim:qualifier="true">true</cimQ:Key>

2552 <cimQ:Description cim:qualifier="true">

2553 An AssociatingComponent represents one Component is associated with the
2554 component referenced as AssociatedComponent.

2555 </cimQ:Description>

2556 **B.4.3.2.3 AssocMaintained**

2557 <cimQ:Description cim:qualifier="true">

2558 Boolean indicating whether the association is maintained.

2559 </cimQ:Description>

2560 **B.4.3.2.4 WhenAssociated**

2561 <cimQ:Description cim:qualifier="true">

2562 The point in time that the Components were associated.

2563 </cimQ:Description>

ANNEX C (Informative)

Collation Optimization Available to Implementers

2568 **C.1 Sorting with Limited Character Set**

2569 The character set permitted in CIM identifiers is limited to

- 2570 - U+0030..U+0039 (digits 0-9)
- 2571 - U+0041..U+005A (alphabetics A-Z)
- 2572 - U+0061..U+007A (alphabetics a-z)
- 2573 - U+0052 (underscore)
- 2574 - U+0080..U+FFEF (the rest of Unicode)

2575 The intention of the preferred collation is that the property name identifier strings should be ordered by a
2576 binary sorting of big-endian UCS-2 representation of the characters. For example, ABC, ABc, and AbC
2577 sort in this order because

2578 ABC 00 41 00 42 00 43

2579 ABc 00 41 00 42 00 63

2580 AbC 00 41 00 62 00 43

2581 (and accented As sort in order by Unicode number)

2582 If the identifiers of a MOF do not use the full range of characters, optimization can be applied to the
2583 sorting of property identifiers. Specifically, a simple approach exists for the commonly observed case
2584 where MOF characters are limited to ASCII-7, i.e., character values < 0x7F.

2585 Most or all MOFs that you will encounter contain only (a subset of) ASCII-7 characters, that is, characters
2586 in the range 0x00 to 0x7F. For such MOFs, it is not necessary to cast the property identifier strings into
2587 Unicode representation at all. If all the characters are ASCII-7, then the strings can be sorted as simple
2588 one-byte sequences.

2589 If the internal representation of character strings (in a CIMOM, protocol adapter, or client application, etc.)
2590 is UTF-8, note that ASCII-7 characters are represented unmodified in UTF-8.

2591 Suggested algorithm: Pre-scan the MOF, or at least the set of property identifiers, for characters > 0x7F.
2592 If the set contains no such characters > 0x7F, then sort the strings as simple one-byte octet strings.

2593 This case can be used for all currently published DMTF MOFs.

2594 **C.2 Note of Caution Concerning Collation**

2595 The default Unicode Collation Algorithm orders properties differently from the current DMTF practice. All
2596 published CIM XML schemata order properties as described here (sorting by Unicode value). The UCA,
2597 by default, uses a different ordering even for the subset of ASCII-7 characters: lower case characters sort
2598 before upper case characters. This results in differences in some cases, where, for example, "CPU" and
2599 "Caption" appear in one order in the published XSD files but sort in the other order using the UCA.

2600 The intention of the preferred algorithm is to retain current DMTF practice. Properties in XSDs will
2601 continue to be in upper-before-lower order. Some existing WS-Man service implementations may be
2602 using the default Unicode Collation Algorithm. These implementations will have to become compatible
2603 with existing practice.

2604

ANNEX D (Informative)

Example Schema With Qualifier Annotations

2605
2606
2607
2608

2609 Following is an example of an XML schema text file that complies with the requirements of this
2610 specification and contains qualifier metadata in annotation elements. The class in the example is
2611 CIM_ComputerSystem, which contains a variety of datatypes and qualifier types.

2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by WBEM Solutions, Inc. SDKPro 3-->
<!--
Example XSD in WS-CIM format with qualifier information added
in annotations, so that complete CIM schema information may be
available to client management applications.

V16 RBL 20110116
- Example XSD produced by WSI from Schema v2.27 contents.
  Indentation added with XMLLint for readability.
V17 RBL 20110201
- Remove carriage returns completely.
-->
<xs:schema xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
xmlns:class="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ComputerSystem"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
elementFormDefault="qualified" targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/CIM_ComputerSystem">
  <xs:import namespace="http://schemas.dmtf.org/wbem/wscim/1/common"
schemaLocation="http://schemas.dmtf.org/wbem/wscim/1/common.xsd"/>
  <xs:element name="NameFormat" nillable="true">
    <xs:annotation>
      <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
        <cim:Qualifier name="Override" type="cim:cimString">NameFormat</cim:Qualifier>
        <cim:Qualifier name="Description" type="cim:cimString">The ComputerSystem object and its
derivatives are Top Level Objects of CIM. They provide the scope for numerous components. Having
unique System keys is required. The NameFormat property identifies how the ComputerSystem Name is
generated. The NameFormat ValueMap qualifier defines the various mechanisms for assigning the
name. Note that another name can be assigned and used for the ComputerSystem that better suit a
business, using the inherited ElementName property.</cim:Qualifier>
        <cim:Qualifier name="MaxLen" type="cim:cimUnsignedInt">64</cim:Qualifier>
        <cim:Qualifier name="Values" type="cim:cimString" valuemap="Other" xsi:nil="true"/>
        <cim:Qualifier name="Values" type="cim:cimString" valuemap="IP" xsi:nil="true"/>
        <cim:Qualifier name="Values" type="cim:cimString" valuemap="Dial" xsi:nil="true"/>
        <cim:Qualifier name="Values" type="cim:cimString" valuemap="HID" xsi:nil="true"/>
        <cim:Qualifier name="Values" type="cim:cimString" valuemap="NWA" xsi:nil="true"/>
        <cim:Qualifier name="Values" type="cim:cimString" valuemap="HWA" xsi:nil="true"/>
        <cim:Qualifier name="Values" type="cim:cimString" valuemap="X25" xsi:nil="true"/>
        <cim:Qualifier name="Values" type="cim:cimString" valuemap="ISDN" xsi:nil="true"/>
        <cim:Qualifier name="Values" type="cim:cimString" valuemap="IPX" xsi:nil="true"/>
        <cim:Qualifier name="Values" type="cim:cimString" valuemap="DCC" xsi:nil="true"/>
      </xs:appinfo>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

```

2657     <cim:Qualifier name="Values" type="cim:cimString" valuemap="ICD" xsi:nil="true"/>
2658     <cim:Qualifier name="Values" type="cim:cimString" valuemap="E.164" xsi:nil="true"/>
2659     <cim:Qualifier name="Values" type="cim:cimString" valuemap="SNA" xsi:nil="true"/>
2660     <cim:Qualifier name="Values" type="cim:cimString" valuemap="OID/OSI" xsi:nil="true"/>
2661     <cim:Qualifier name="Values" type="cim:cimString" valuemap="WWN" xsi:nil="true"/>
2662     <cim:Qualifier name="Values" type="cim:cimString" valuemap="NAA" xsi:nil="true"/>
2663     <cim:Classorigin name="CIM_ComputerSystem"/>
2664   </xs:appinfo>
2665 </xs:annotation>
2666 <xs:complexType>
2667   <xs:simpleContent>
2668     <xs:restriction base="cim:cimString">
2669       <xs:maxLength value="64"/>
2670       <xs:enumeration value="Other"/>
2671       <xs:enumeration value="IP"/>
2672       <xs:enumeration value="Dial"/>
2673       <xs:enumeration value="HID"/>
2674       <xs:enumeration value="NWA"/>
2675       <xs:enumeration value="HWA"/>
2676       <xs:enumeration value="X25"/>
2677       <xs:enumeration value="ISDN"/>
2678       <xs:enumeration value="IPX"/>
2679       <xs:enumeration value="DCC"/>
2680       <xs:enumeration value="ICD"/>
2681       <xs:enumeration value="E.164"/>
2682       <xs:enumeration value="SNA"/>
2683       <xs:enumeration value="OID/OSI"/>
2684       <xs:enumeration value="WWN"/>
2685       <xs:enumeration value="NAA"/>
2686       <xs:anyAttribute namespace="##any" processContents="lax"/>
2687     </xs:restriction>
2688   </xs:simpleContent>
2689 </xs:complexType>
2690 </xs:element>
2691 <xs:element name="Dedicated" nillable="true">
2692   <xs:annotation>
2693     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
2694       <cim:Qualifier name="Description" type="cim:cimString">Enumeration indicating the
2695       purpose(s) to which the ComputerSystem is dedicated, if any, and what functionality is provided.
2696       For example, one could specify that the System is dedicated to "Print" (value=11) or acts as a
2697       "Hub" (value=8).
2698       Also, one could indicate that this is a general purpose system by indicating 'Not Dedicated'
2699       (value=0) but that it also hosts 'Print' (value=11) or mobile phone 'Mobile User Device'
2700       (value=17) services.
2701       A clarification is needed with respect to the value 17 ("Mobile User Device"). An example of a
2702       dedicated user device is a mobile phone or a barcode scanner in a store that communicates via
2703       radio frequency. These systems are quite limited in functionality and programmability, and are
2704       not considered 'general purpose' computing platforms. Alternately, an example of a mobile system
2705       that is 'general purpose' (i.e., is NOT dedicated) is a hand-held computer. Although limited in
2706       its programmability, new software can be downloaded and its functionality expanded by the user.
2707       A value of "Management" indicates this instance is dedicated to hosting system management
2708       software.
2709       A value of "Management Controller" indicates this instance represents specialized hardware
2710       dedicated to systems management (i.e., a Baseboard Management Controller (BMC) or service
2711       processor).
2712       The management scope of a "Management Controller" is typically a single managed system in which
2713       it is contained.

```

2714 A value of "Chassis Manager" indicates this instance represents a system dedicated to management
2715 of a blade chassis and its contained devices. This value would be used to represent a Shelf
2716 Controller. A "Chassis Manager" is an aggregation point for management and may rely on
2717 subordinate management controllers for the management of constituent parts. A value of "Host-
2718 based RAID Controller" indicates this instance represents a RAID storage controller contained
2719 within a host computer. A value of "Storage Device Enclosure" indicates this instance represents
2720 an enclosure that contains storage devices. A "Virtual Tape Library" is the emulation of a tape
2721 library by a Virtual Library System. A "Virtual Library System" uses disk storage to emulate tape
2722 libraries. A "FC Switch" indicates this instance is dedicated to switching layer 2 fibre channel
2723 frames. An "Ethernet Switch" indicates this instance is dedicated to switching layer 2 ethernet
2724 frames.</cim:Qualifier>

2725 <cim:Qualifier name="ArrayType" type="cim:cimString">Indexed</cim:Qualifier>

2726 <cim:Qualifier name="MappingStrings" type="cim:cimString">MIB.IETF|MIB-
2727 II.sysServices</cim:Qualifier>

2728 <cim:Qualifier name="MappingStrings" type="cim:cimString">FC-GS.INCITS-T11 | Platform |
2729 PlatformType</cim:Qualifier>

2730 <cim:Qualifier name="ModelCorrespondence"
2731 type="cim:cimString">CIM_ComputerSystem.OtherDedicatedDescriptions</cim:Qualifier>

2732 <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Not
2733 Dedicated</cim:Qualifier>

2734 <cim:Qualifier name="Values" type="cim:cimString" valuemap="1">Unknown</cim:Qualifier>

2735 <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Other</cim:Qualifier>

2736 <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Storage</cim:Qualifier>

2737 <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Router</cim:Qualifier>

2738 <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">Switch</cim:Qualifier>

2739 <cim:Qualifier name="Values" type="cim:cimString" valuemap="6">Layer 3
2740 Switch</cim:Qualifier>

2741 <cim:Qualifier name="Values" type="cim:cimString" valuemap="7">Central Office
2742 Switch</cim:Qualifier>

2743 <cim:Qualifier name="Values" type="cim:cimString" valuemap="8">Hub</cim:Qualifier>

2744 <cim:Qualifier name="Values" type="cim:cimString" valuemap="9">Access
2745 Server</cim:Qualifier>

2746 <cim:Qualifier name="Values" type="cim:cimString" valuemap="10">Firewall</cim:Qualifier>

2747 <cim:Qualifier name="Values" type="cim:cimString" valuemap="11">Print</cim:Qualifier>

2748 <cim:Qualifier name="Values" type="cim:cimString" valuemap="12">I/O</cim:Qualifier>

2749 <cim:Qualifier name="Values" type="cim:cimString" valuemap="13">Web
2750 Caching</cim:Qualifier>

2751 <cim:Qualifier name="Values" type="cim:cimString"
2752 valuemap="14">Management</cim:Qualifier>

2753 <cim:Qualifier name="Values" type="cim:cimString" valuemap="15">Block
2754 Server</cim:Qualifier>

2755 <cim:Qualifier name="Values" type="cim:cimString" valuemap="16">File
2756 Server</cim:Qualifier>

2757 <cim:Qualifier name="Values" type="cim:cimString" valuemap="17">Mobile User
2758 Device</cim:Qualifier>

2759 <cim:Qualifier name="Values" type="cim:cimString" valuemap="18">Repeater</cim:Qualifier>

2760 <cim:Qualifier name="Values" type="cim:cimString"
2761 valuemap="19">Bridge/Extender</cim:Qualifier>

2762 <cim:Qualifier name="Values" type="cim:cimString" valuemap="20">Gateway</cim:Qualifier>

2763 <cim:Qualifier name="Values" type="cim:cimString" valuemap="21">Storage
2764 Virtualizer</cim:Qualifier>

2765 <cim:Qualifier name="Values" type="cim:cimString" valuemap="22">Media
2766 Library</cim:Qualifier>

2767 <cim:Qualifier name="Values" type="cim:cimString"
2768 valuemap="23">ExtenderNode</cim:Qualifier>

2769 <cim:Qualifier name="Values" type="cim:cimString" valuemap="24">NAS Head</cim:Qualifier>

2770 <cim:Qualifier name="Values" type="cim:cimString" valuemap="25">Self-contained
2771 NAS</cim:Qualifier>

2772 <cim:Qualifier name="Values" type="cim:cimString" valuemap="26">UPS</cim:Qualifier>

2773 <cim:Qualifier name="Values" type="cim:cimString" valuemap="27">IP Phone</cim:Qualifier>

```

2774     <cim:Qualifier name="Values" type="cim:cimString" valuemap="28">Management
2775 Controller</cim:Qualifier>
2776     <cim:Qualifier name="Values" type="cim:cimString" valuemap="29">Chassis
2777 Manager</cim:Qualifier>
2778     <cim:Qualifier name="Values" type="cim:cimString" valuemap="30">Host-based RAID
2779 controller</cim:Qualifier>
2780     <cim:Qualifier name="Values" type="cim:cimString" valuemap="31">Storage Device
2781 Enclosure</cim:Qualifier>
2782     <cim:Qualifier name="Values" type="cim:cimString" valuemap="32">Desktop</cim:Qualifier>
2783     <cim:Qualifier name="Values" type="cim:cimString" valuemap="33">Laptop</cim:Qualifier>
2784     <cim:Qualifier name="Values" type="cim:cimString" valuemap="34">Virtual Tape
2785 Library</cim:Qualifier>
2786     <cim:Qualifier name="Values" type="cim:cimString" valuemap="35">Virtual Library
2787 System</cim:Qualifier>
2788     <cim:Qualifier name="Values" type="cim:cimString" valuemap="36">Network PC/Thin
2789 Client</cim:Qualifier>
2790     <cim:Qualifier name="Values" type="cim:cimString" valuemap="37">FC Switch</cim:Qualifier>
2791     <cim:Qualifier name="Values" type="cim:cimString" valuemap="38">Ethernet
2792 Switch</cim:Qualifier>
2793     <cim:Qualifier name="Values" type="cim:cimString" valuemap="..">DMTF
2794 Reserved</cim:Qualifier>
2795     <cim:Qualifier name="Values" type="cim:cimString" valuemap="32568..65535">Vendor
2796 Reserved</cim:Qualifier>
2797     <cim:Classorigin name="CIM_ComputerSystem"/>
2798 </xs:appinfo>
2799 </xs:annotation>
2800 <xs:complexType>
2801   <xs:simpleContent>
2802     <xs:restriction base="cim:cimAnySimpleType">
2803       <xs:simpleType>
2804         <xs:union>
2805           <xs:simpleType>
2806             <xs:restriction base="xs:unsignedShort">
2807               <xs:enumeration value="0"/>
2808               <xs:enumeration value="1"/>
2809               <xs:enumeration value="2"/>
2810               <xs:enumeration value="3"/>
2811               <xs:enumeration value="4"/>
2812               <xs:enumeration value="5"/>
2813               <xs:enumeration value="6"/>
2814               <xs:enumeration value="7"/>
2815               <xs:enumeration value="8"/>
2816               <xs:enumeration value="9"/>
2817               <xs:enumeration value="10"/>
2818               <xs:enumeration value="11"/>
2819               <xs:enumeration value="12"/>
2820               <xs:enumeration value="13"/>
2821               <xs:enumeration value="14"/>
2822               <xs:enumeration value="15"/>
2823               <xs:enumeration value="16"/>
2824               <xs:enumeration value="17"/>
2825               <xs:enumeration value="18"/>
2826               <xs:enumeration value="19"/>
2827               <xs:enumeration value="20"/>
2828               <xs:enumeration value="21"/>
2829               <xs:enumeration value="22"/>

```

```

2830         <xs:enumeration value="23"/>
2831         <xs:enumeration value="24"/>
2832         <xs:enumeration value="25"/>
2833         <xs:enumeration value="26"/>
2834         <xs:enumeration value="27"/>
2835         <xs:enumeration value="28"/>
2836         <xs:enumeration value="29"/>
2837         <xs:enumeration value="30"/>
2838         <xs:enumeration value="31"/>
2839         <xs:enumeration value="32"/>
2840         <xs:enumeration value="33"/>
2841         <xs:enumeration value="34"/>
2842         <xs:enumeration value="35"/>
2843         <xs:enumeration value="36"/>
2844         <xs:enumeration value="37"/>
2845         <xs:enumeration value="38"/>
2846     </xs:restriction>
2847 </xs:simpleType>
2848 <xs:simpleType>
2849     <xs:union>
2850         <xs:simpleType>
2851             <xs:restriction base="xs:unsignedShort">
2852                 <xs:minInclusive value="39"/>
2853                 <xs:maxInclusive value="32567"/>
2854             </xs:restriction>
2855         </xs:simpleType>
2856     </xs:union>
2857 </xs:simpleType>
2858 <xs:simpleType>
2859     <xs:restriction base="xs:unsignedShort">
2860         <xs:minInclusive value="32568"/>
2861         <xs:maxInclusive value="65535"/>
2862     </xs:restriction>
2863 </xs:simpleType>
2864 </xs:union>
2865 </xs:simpleType>
2866     <xs:anyAttribute namespace="##any" processContents="lax"/>
2867 </xs:restriction>
2868 </xs:simpleContent>
2869 </xs:complexType>
2870 </xs:element>
2871 <xs:element name="OtherDedicatedDescriptions" nillable="true" type="cim:cimString">
2872     <xs:annotation>
2873         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
2874             <cim:Qualifier name="Description" type="cim:cimString">A string describing how or why the
2875 system is dedicated when the Dedicated array includes the value 2, "Other".</cim:Qualifier>
2876             <cim:Qualifier name="ArrayType" type="cim:cimString">Indexed</cim:Qualifier>
2877             <cim:Qualifier name="ModelCorrespondence"
2878 type="cim:cimString">CIM_ComputerSystem.Dedicated</cim:Qualifier>
2879             <cim:Classorigin name="CIM_ComputerSystem"/>
2880         </xs:appinfo>
2881     </xs:annotation>
2882 </xs:element>
2883 <xs:element name="ResetCapability" nillable="true">

```

```

2884     <xs:annotation>
2885         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
2886             <cim:Qualifier name="Description" type="cim:cimString">If enabled (value = 4), the
2887 ComputerSystem can be reset via hardware (e.g. the power and reset buttons). If disabled (value =
2888 3), hardware reset is not allowed. In addition to Enabled and Disabled, other Values for the
2889 property are also defined - "Not Implemented" (5), "Other" (1) and "Unknown" (2).</cim:Qualifier>
2890             <cim:Qualifier name="MappingStrings" type="cim:cimString">MIF.DMTF|System Hardware
2891 Security|001.4</cim:Qualifier>
2892             <cim:Qualifier name="Values" type="cim:cimString" valuemap="1">Other</cim:Qualifier>
2893             <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Unknown</cim:Qualifier>
2894             <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Disabled</cim:Qualifier>
2895             <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Enabled</cim:Qualifier>
2896             <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">Not
2897 Implemented</cim:Qualifier>
2898             <cim:Classorigin name="CIM_ComputerSystem"/>
2899         </xs:appinfo>
2900     </xs:annotation>
2901     <xs:complexType>
2902         <xs:simpleContent>
2903             <xs:restriction base="cim:cimUnsignedShort">
2904                 <xs:enumeration value="1"/>
2905                 <xs:enumeration value="2"/>
2906                 <xs:enumeration value="3"/>
2907                 <xs:enumeration value="4"/>
2908                 <xs:enumeration value="5"/>
2909                 <xs:anyAttribute namespace="##any" processContents="lax"/>
2910             </xs:restriction>
2911         </xs:simpleContent>
2912     </xs:complexType>
2913 </xs:element>
2914 <xs:element name="PowerManagementCapabilities" nillable="true">
2915     <xs:annotation>
2916         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
2917             <cim:Qualifier name="Deprecated"
2918 type="cim:cimString">CIM_PowerManagementCapabilities.PowerCapabilities</cim:Qualifier>
2919             <cim:Qualifier name="Description" type="cim:cimString">An enumerated array describing the
2920 power management capabilities of the ComputerSystem. The use of this property has been
2921 deprecated. Instead, the Power Capabilites property in an associated PowerManagement Capabilities
2922 class should be used.</cim:Qualifier>
2923             <cim:Qualifier name="MappingStrings" type="cim:cimString">MIF.DMTF|System Power
2924 Controls|001.2</cim:Qualifier>
2925             <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Unknown</cim:Qualifier>
2926             <cim:Qualifier name="Values" type="cim:cimString" valuemap="1">Not
2927 Supported</cim:Qualifier>
2928             <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Disabled</cim:Qualifier>
2929             <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Enabled</cim:Qualifier>
2930             <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Power Saving Modes Entered
2931 Automatically</cim:Qualifier>
2932             <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">Power State
2933 Settable</cim:Qualifier>
2934             <cim:Qualifier name="Values" type="cim:cimString" valuemap="6">Power Cycling
2935 Supported</cim:Qualifier>
2936             <cim:Qualifier name="Values" type="cim:cimString" valuemap="7">Timed Power On
2937 Supported</cim:Qualifier>
2938             <cim:Classorigin name="CIM_ComputerSystem"/>
2939         </xs:appinfo>
2940     </xs:annotation>

```

```

2941 <xs:complexType>
2942   <xs:simpleContent>
2943     <xs:restriction base="cim:cimUnsignedShort">
2944       <xs:enumeration value="0"/>
2945       <xs:enumeration value="1"/>
2946       <xs:enumeration value="2"/>
2947       <xs:enumeration value="3"/>
2948       <xs:enumeration value="4"/>
2949       <xs:enumeration value="5"/>
2950       <xs:enumeration value="6"/>
2951       <xs:enumeration value="7"/>
2952       <xs:anyAttribute namespace="##any" processContents="lax"/>
2953     </xs:restriction>
2954   </xs:simpleContent>
2955 </xs:complexType>
2956 </xs:element>
2957 <xs:element name="CreationClassName">
2958   <xs:annotation>
2959     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
2960       <cim:Qualifier name="Key" type="cim:cimBoolean">true</cim:Qualifier>
2961       <cim:Qualifier name="Description" type="cim:cimString">CreationClassName indicates the
2962 name of the class or the subclass used in the creation of an instance. When used with the other
2963 key properties of this class, this property allows all instances of this class and its subclasses
2964 to be uniquely identified.</cim:Qualifier>
2965       <cim:Qualifier name="MaxLen" type="cim:cimUnsignedInt">256</cim:Qualifier>
2966       <cim:Classorigin name="CIM_System"/>
2967     </xs:appinfo>
2968   </xs:annotation>
2969   <xs:complexType>
2970     <xs:simpleContent>
2971       <xs:restriction base="cim:cimString">
2972         <xs:maxLength value="256"/>
2973         <xs:anyAttribute namespace="##any" processContents="lax"/>
2974       </xs:restriction>
2975     </xs:simpleContent>
2976   </xs:complexType>
2977 </xs:element>
2978 <xs:element name="Name">
2979   <xs:annotation>
2980     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
2981       <cim:Qualifier name="Key" type="cim:cimBoolean">true</cim:Qualifier>
2982       <cim:Qualifier name="Override" type="cim:cimString">Name</cim:Qualifier>
2983       <cim:Qualifier name="Description" type="cim:cimString">The inherited Name serves as the
2984 key of a System instance in an enterprise environment.</cim:Qualifier>
2985       <cim:Qualifier name="MaxLen" type="cim:cimUnsignedInt">256</cim:Qualifier>
2986       <cim:Classorigin name="CIM_System"/>
2987     </xs:appinfo>
2988   </xs:annotation>
2989   <xs:complexType>
2990     <xs:simpleContent>
2991       <xs:restriction base="cim:cimString">
2992         <xs:maxLength value="256"/>
2993         <xs:anyAttribute namespace="##any" processContents="lax"/>
2994       </xs:restriction>

```



```

2995     </xs:simpleContent>
2996   </xs:complexType>
2997 </xs:element>
2998 <xs:element name="PrimaryOwnerName" nillable="true">
2999   <xs:annotation>
3000     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3001       <cim:Qualifier name="Write" type="cim:cimBoolean">true</cim:Qualifier>
3002       <cim:Qualifier name="Description" type="cim:cimString">The name of the primary system
3003 owner. The system owner is the primary user of the system.</cim:Qualifier>
3004       <cim:Qualifier name="MaxLen" type="cim:cimUnsignedInt">64</cim:Qualifier>
3005       <cim:Qualifier name="MappingStrings" type="cim:cimString">MIF.DMTF|General
3006 Information|001.3</cim:Qualifier>
3007       <cim:Classorigin name="CIM_System"/>
3008     </xs:appinfo>
3009   </xs:annotation>
3010   <xs:complexType>
3011     <xs:simpleContent>
3012       <xs:restriction base="cim:cimString">
3013         <xs:maxLength value="64"/>
3014         <xs:anyAttribute namespace="##any" processContents="lax"/>
3015       </xs:restriction>
3016     </xs:simpleContent>
3017   </xs:complexType>
3018 </xs:element>
3019 <xs:element name="PrimaryOwnerContact" nillable="true">
3020   <xs:annotation>
3021     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3022       <cim:Qualifier name="Write" type="cim:cimBoolean">true</cim:Qualifier>
3023       <cim:Qualifier name="Description" type="cim:cimString">A string that provides information
3024 on how the primary system owner can be reached (for example, phone number, e-mail address, and so
3025 on).</cim:Qualifier>
3026       <cim:Qualifier name="MaxLen" type="cim:cimUnsignedInt">256</cim:Qualifier>
3027       <cim:Qualifier name="MappingStrings" type="cim:cimString">MIF.DMTF|General
3028 Information|001.4</cim:Qualifier>
3029       <cim:Classorigin name="CIM_System"/>
3030     </xs:appinfo>
3031   </xs:annotation>
3032   <xs:complexType>
3033     <xs:simpleContent>
3034       <xs:restriction base="cim:cimString">
3035         <xs:maxLength value="256"/>
3036         <xs:anyAttribute namespace="##any" processContents="lax"/>
3037       </xs:restriction>
3038     </xs:simpleContent>
3039   </xs:complexType>
3040 </xs:element>
3041 <xs:element name="Roles" nillable="true" type="cim:cimString">
3042   <xs:annotation>
3043     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3044       <cim:Qualifier name="Write" type="cim:cimBoolean">true</cim:Qualifier>
3045       <cim:Qualifier name="Description" type="cim:cimString">An array (bag) of strings that
3046 specifies the administrator -defined roles this System plays in the managed environment. Examples
3047 might be 'Building 8 print server' or 'Boise user directories'. A single system may perform
3048 multiple roles.
3049 Note that the instrumentation view of the 'roles' of a System is defined by instantiating a
3050 specific subclass of System, or by properties in a subclass, or both. For example, the purpose of

```

```

3051 a ComputerSystem is defined using the Dedicated and OtherDedicatedDescription
3052 properties.</cim:Qualifier>
3053     <cim:Classorigin name="CIM_System"/>
3054     </xs:appinfo>
3055     </xs:annotation>
3056     </xs:element>
3057     <xs:element name="OtherIdentifyingInfo" nillable="true">
3058     <xs:annotation>
3059     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3060     <cim:Qualifier name="Description" type="cim:cimString">OtherIdentifyingInfo captures
3061 additional data, beyond System Name information, that could be used to identify a ComputerSystem.
3062 One example would be to hold the Fibre Channel World-Wide Name (WWN) of a node. Note that if only
3063 the Fibre Channel name is available and is unique (able to be used as the System key), then this
3064 property would be NULL and the WWN would become the System key, its data placed in the Name
3065 property.</cim:Qualifier>
3066     <cim:Qualifier name="ArrayType" type="cim:cimString">Indexed</cim:Qualifier>
3067     <cim:Qualifier name="MaxLen" type="cim:cimUnsignedInt">256</cim:Qualifier>
3068     <cim:Qualifier name="ModelCorrespondence"
3069 type="cim:cimString">CIM_System.IdentifyingDescriptions</cim:Qualifier>
3070     <cim:Classorigin name="CIM_System"/>
3071     </xs:appinfo>
3072     </xs:annotation>
3073     <xs:complexType>
3074     <xs:simpleContent>
3075     <xs:restriction base="cim:cimString">
3076     <xs:maxLength value="256"/>
3077     <xs:anyAttribute namespace="##any" processContents="lax"/>
3078     </xs:restriction>
3079     </xs:simpleContent>
3080     </xs:complexType>
3081     </xs:element>
3082     <xs:element name="IdentifyingDescriptions" nillable="true" type="cim:cimString">
3083     <xs:annotation>
3084     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3085     <cim:Qualifier name="Description" type="cim:cimString">An array of free-form strings
3086 providing explanations and details behind the entries in the OtherIdentifying Info array. Note,
3087 each entry of this array is related to the entry in OtherIdentifyingInfo that is located at the
3088 same index.</cim:Qualifier>
3089     <cim:Qualifier name="ArrayType" type="cim:cimString">Indexed</cim:Qualifier>
3090     <cim:Qualifier name="ModelCorrespondence"
3091 type="cim:cimString">CIM_System.OtherIdentifyingInfo</cim:Qualifier>
3092     <cim:Classorigin name="CIM_System"/>
3093     </xs:appinfo>
3094     </xs:annotation>
3095     </xs:element>
3096     <xs:element name="EnabledState" nillable="true">
3097     <xs:annotation>
3098     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3099     <cim:Qualifier name="Description" type="cim:cimString">EnabledState is an integer
3100 enumeration that indicates the enabled and disabled states of an element. It can also indicate
3101 the transitions between these requested states. For example, shutting down (value=4) and starting
3102 (value=10) are transient states between enabled and disabled. The following text briefly
3103 summarizes the various enabled and disabled states:
3104 Enabled (2) indicates that the element is or could be executing commands, will process any queued
3105 commands, and queues new requests.
3106 Disabled (3) indicates that the element will not execute commands and will drop any new requests.
3107 Shutting Down (4) indicates that the element is in the process of going to a Disabled state.

```

```

3108 Not Applicable (5) indicates the element does not support being enabled or disabled.
3109 Enabled but Offline (6) indicates that the element might be completing commands, and will drop
3110 any new requests.
3111 Test (7) indicates that the element is in a test state.
3112 Deferred (8) indicates that the element might be completing commands, but will queue any new
3113 requests.
3114 Quiesce (9) indicates that the element is enabled but in a restricted mode.
3115 Starting (10) indicates that the element is in the process of going to an Enabled state. New
3116 requests are queued.</cim:Qualifier>
3117     <cim:Qualifier name="ModelCorrespondence"
3118 type="cim:cimString">CIM_EnabledLogicalElement.OtherEnabledState</cim:Qualifier>
3119     <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Unknown</cim:Qualifier>
3120     <cim:Qualifier name="Values" type="cim:cimString" valuemap="1">Other</cim:Qualifier>
3121     <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Enabled</cim:Qualifier>
3122     <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Disabled</cim:Qualifier>
3123     <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Shutting
3124 Down</cim:Qualifier>
3125     <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">Not
3126 Applicable</cim:Qualifier>
3127     <cim:Qualifier name="Values" type="cim:cimString" valuemap="6">Enabled but
3128 Offline</cim:Qualifier>
3129     <cim:Qualifier name="Values" type="cim:cimString" valuemap="7">In Test</cim:Qualifier>
3130     <cim:Qualifier name="Values" type="cim:cimString" valuemap="8">Deferred</cim:Qualifier>
3131     <cim:Qualifier name="Values" type="cim:cimString" valuemap="9">Quiesce</cim:Qualifier>
3132     <cim:Qualifier name="Values" type="cim:cimString" valuemap="10">Starting</cim:Qualifier>
3133     <cim:Qualifier name="Values" type="cim:cimString" valuemap="11..32767">DMTF
3134 Reserved</cim:Qualifier>
3135     <cim:Qualifier name="Values" type="cim:cimString" valuemap="32768..65535">Vendor
3136 Reserved</cim:Qualifier>
3137     <cim:Classorigin name="CIM_EnabledLogicalElement" />
3138     <cim:Defaultvalue type="cim:cimUnsignedShort">5</cim:Defaultvalue>
3139 </xs:appinfo>
3140 </xs:annotation>
3141 <xs:complexType>
3142   <xs:simpleContent>
3143     <xs:restriction base="cim:cimAnySimpleType">
3144       <xs:simpleType>
3145         <xs:union>
3146           <xs:simpleType>
3147             <xs:restriction base="xs:unsignedShort">
3148               <xs:enumeration value="0"/>
3149               <xs:enumeration value="1"/>
3150               <xs:enumeration value="2"/>
3151               <xs:enumeration value="3"/>
3152               <xs:enumeration value="4"/>
3153               <xs:enumeration value="5"/>
3154               <xs:enumeration value="6"/>
3155               <xs:enumeration value="7"/>
3156               <xs:enumeration value="8"/>
3157               <xs:enumeration value="9"/>
3158               <xs:enumeration value="10"/>
3159             </xs:restriction>
3160           </xs:simpleType>
3161         </xs:union>
3162       </xs:simpleType>
3163     </xs:restriction>

```

```

3164         <xs:maxInclusive value="32767"/>
3165     </xs:restriction>
3166 </xs:simpleType>
3167 <xs:simpleType>
3168     <xs:restriction base="xs:unsignedShort">
3169         <xs:minInclusive value="32768"/>
3170         <xs:maxInclusive value="65535"/>
3171     </xs:restriction>
3172 </xs:simpleType>
3173 </xs:union>
3174 </xs:simpleType>
3175 <xs:anyAttribute namespace="##any" processContents="lax"/>
3176 </xs:restriction>
3177 </xs:simpleContent>
3178 </xs:complexType>
3179 </xs:element>
3180 <xs:element name="OtherEnabledState" nillable="true" type="cim:cimString">
3181     <xs:annotation>
3182         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3183             <cim:Qualifier name="Description" type="cim:cimString">A string that describes the
3184 enabled or disabled state of the element when the EnabledState property is set to 1 ("Other").
3185 This property must be set to null when EnabledState is any value other than 1.</cim:Qualifier>
3186             <cim:Qualifier name="ModelCorrespondence"
3187 type="cim:cimString">CIM_EnabledLogicalElement.EnabledState</cim:Qualifier>
3188             <cim:Classorigin name="CIM_EnabledLogicalElement"/>
3189         </xs:appinfo>
3190     </xs:annotation>
3191 </xs:element>
3192 <xs:element name="RequestedState" nillable="true">
3193     <xs:annotation>
3194         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3195             <cim:Qualifier name="Description" type="cim:cimString">RequestedState is an integer
3196 enumeration that indicates the last requested or desired state for the element, irrespective of
3197 the mechanism through which it was requested. The actual state of the element is represented by
3198 EnabledState. This property is provided to compare the last requested and current enabled or
3199 disabled states. Note that when EnabledState is set to 5 ("Not Applicable"), then this property
3200 has no meaning. Refer to the EnabledState property description for explanations of the values in
3201 the RequestedState enumeration.
3202 "Unknown" (0) indicates the last requested state for the element is unknown.
3203 Note that the value "No Change" (5) has been deprecated in lieu of indicating the last requested
3204 state is "Unknown" (0). If the last requested or desired state is unknown, RequestedState should
3205 have the value "Unknown" (0), but may have the value "No Change" (5).Offline (6) indicates that
3206 the element has been requested to transition to the Enabled but Offline EnabledState.
3207 It should be noted that there are two new values in RequestedState that build on the statuses of
3208 EnabledState. These are "Reboot" (10) and "Reset" (11). Reboot refers to doing a "Shut Down" and
3209 then moving to an "Enabled" state. Reset indicates that the element is first "Disabled" and then
3210 "Enabled". The distinction between requesting "Shut Down" and "Disabled" should also be noted.
3211 Shut Down requests an orderly transition to the Disabled state, and might involve removing power,
3212 to completely erase any existing state. The Disabled state requests an immediate disabling of the
3213 element, such that it will not execute or accept any commands or processing requests.
3214
3215 This property is set as the result of a method invocation (such as Start or StopService on
3216 CIM_Service), or can be overridden and defined as WRITEable in a subclass. The method approach is
3217 considered superior to a WRITEable property, because it allows an explicit invocation of the
3218 operation and the return of a result code.
3219
3220 If knowledge of the last RequestedState is not supported for the EnabledLogicalElement, the
3221 property shall be NULL or have the value 12 "Not Applicable".</cim:Qualifier>

```

```

3222     <cim:Qualifier name="ModelCorrespondence"
3223 type="cim:cimString">CIM_EnabledLogicalElement.EnabledState</cim:Qualifier>
3224     <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Unknown</cim:Qualifier>
3225     <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Enabled</cim:Qualifier>
3226     <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Disabled</cim:Qualifier>
3227     <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Shut Down</cim:Qualifier>
3228     <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">No Change</cim:Qualifier>
3229     <cim:Qualifier name="Values" type="cim:cimString" valuemap="6">Offline</cim:Qualifier>
3230     <cim:Qualifier name="Values" type="cim:cimString" valuemap="7">Test</cim:Qualifier>
3231     <cim:Qualifier name="Values" type="cim:cimString" valuemap="8">Deferred</cim:Qualifier>
3232     <cim:Qualifier name="Values" type="cim:cimString" valuemap="9">Quiesce</cim:Qualifier>
3233     <cim:Qualifier name="Values" type="cim:cimString" valuemap="10">Reboot</cim:Qualifier>
3234     <cim:Qualifier name="Values" type="cim:cimString" valuemap="11">Reset</cim:Qualifier>
3235     <cim:Qualifier name="Values" type="cim:cimString" valuemap="12">Not
3236 Applicable</cim:Qualifier>
3237     <cim:Qualifier name="Values" type="cim:cimString" valuemap="..">DMTF
3238 Reserved</cim:Qualifier>
3239     <cim:Qualifier name="Values" type="cim:cimString" valuemap="32768..65535">Vendor
3240 Reserved</cim:Qualifier>
3241     <cim:Classorigin name="CIM_EnabledLogicalElement" />
3242     <cim:Defaultvalue type="cim:cimUnsignedShort">12</cim:Defaultvalue>
3243 </xs:appinfo>
3244 </xs:annotation>
3245 <xs:complexType>
3246   <xs:simpleContent>
3247     <xs:restriction base="cim:cimAnySimpleType">
3248       <xs:simpleType>
3249         <xs:union>
3250           <xs:simpleType>
3251             <xs:restriction base="xs:unsignedShort">
3252               <xs:enumeration value="0" />
3253               <xs:enumeration value="2" />
3254               <xs:enumeration value="3" />
3255               <xs:enumeration value="4" />
3256               <xs:enumeration value="5" />
3257               <xs:enumeration value="6" />
3258               <xs:enumeration value="7" />
3259               <xs:enumeration value="8" />
3260               <xs:enumeration value="9" />
3261               <xs:enumeration value="10" />
3262               <xs:enumeration value="11" />
3263               <xs:enumeration value="12" />
3264             </xs:restriction>
3265           </xs:simpleType>
3266           <xs:simpleType>
3267             <xs:union>
3268               <xs:simpleType>
3269                 <xs:restriction base="xs:unsignedShort">
3270                   <xs:enumeration value="1" />
3271                 </xs:restriction>
3272               </xs:simpleType>
3273               <xs:simpleType>
3274                 <xs:restriction base="xs:unsignedShort">
3275                   <xs:minInclusive value="13" />

```

```

3276         <xs:maxInclusive value="32767"/>
3277         </xs:restriction>
3278         </xs:simpleType>
3279     </xs:union>
3280 </xs:simpleType>
3281 <xs:simpleType>
3282     <xs:restriction base="xs:unsignedShort">
3283         <xs:minInclusive value="32768"/>
3284         <xs:maxInclusive value="65535"/>
3285     </xs:restriction>
3286 </xs:simpleType>
3287 </xs:union>
3288 </xs:simpleType>
3289     <xs:anyAttribute namespace="##any" processContents="lax"/>
3290 </xs:restriction>
3291 </xs:simpleContent>
3292 </xs:complexType>
3293 </xs:element>
3294 <xs:element name="EnabledDefault" nillable="true">
3295     <xs:annotation>
3296         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3297             <cim:Qualifier name="Write" type="cim:cimBoolean">true</cim:Qualifier>
3298             <cim:Qualifier name="Description" type="cim:cimString">An enumerated value indicating an
3299 administrator's default or startup configuration for the Enabled State of an element. By default,
3300 the element is "Enabled" (value=2).</cim:Qualifier>
3301             <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Enabled</cim:Qualifier>
3302             <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Disabled</cim:Qualifier>
3303             <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">Not
3304 Applicable</cim:Qualifier>
3305             <cim:Qualifier name="Values" type="cim:cimString" valuemap="6">Enabled but
3306 Offline</cim:Qualifier>
3307             <cim:Qualifier name="Values" type="cim:cimString" valuemap="7">No Default</cim:Qualifier>
3308             <cim:Qualifier name="Values" type="cim:cimString" valuemap="9">Quiesce</cim:Qualifier>
3309             <cim:Qualifier name="Values" type="cim:cimString" valuemap="..">DMTF
3310 Reserved</cim:Qualifier>
3311             <cim:Qualifier name="Values" type="cim:cimString" valuemap="32768..65535">Vendor
3312 Reserved</cim:Qualifier>
3313             <cim:Classorigin name="CIM_EnabledLogicalElement"/>
3314             <cim:Defaultvalue type="cim:cimUnsignedShort">2</cim:Defaultvalue>
3315         </xs:appinfo>
3316     </xs:annotation>
3317 </xs:complexType>
3318 <xs:simpleContent>
3319     <xs:restriction base="cim:cimAnySimpleType">
3320         <xs:simpleType>
3321             <xs:union>
3322                 <xs:simpleType>
3323                     <xs:restriction base="xs:unsignedShort">
3324                         <xs:enumeration value="2"/>
3325                         <xs:enumeration value="3"/>
3326                         <xs:enumeration value="5"/>
3327                         <xs:enumeration value="6"/>
3328                         <xs:enumeration value="7"/>
3329                         <xs:enumeration value="9"/>
3330                     </xs:restriction>

```

```

3331     </xs:simpleType>
3332     <xs:simpleType>
3333         <xs:union>
3334             <xs:simpleType>
3335                 <xs:restriction base="xs:unsignedShort">
3336                     <xs:maxInclusive value="1"/>
3337                 </xs:restriction>
3338             </xs:simpleType>
3339             <xs:simpleType>
3340                 <xs:restriction base="xs:unsignedShort">
3341                     <xs:enumeration value="4"/>
3342                 </xs:restriction>
3343             </xs:simpleType>
3344             <xs:simpleType>
3345                 <xs:restriction base="xs:unsignedShort">
3346                     <xs:enumeration value="8"/>
3347                 </xs:restriction>
3348             </xs:simpleType>
3349             <xs:simpleType>
3350                 <xs:restriction base="xs:unsignedShort">
3351                     <xs:minInclusive value="10"/>
3352                     <xs:maxInclusive value="32767"/>
3353                 </xs:restriction>
3354             </xs:simpleType>
3355         </xs:union>
3356     </xs:simpleType>
3357     <xs:simpleType>
3358         <xs:restriction base="xs:unsignedShort">
3359             <xs:minInclusive value="32768"/>
3360             <xs:maxInclusive value="65535"/>
3361         </xs:restriction>
3362     </xs:simpleType>
3363 </xs:union>
3364 </xs:simpleType>
3365 <xs:anyAttribute namespace="##any" processContents="lax"/>
3366 </xs:restriction>
3367 </xs:simpleContent>
3368 </xs:complexType>
3369 </xs:element>
3370 <xs:element name="TimeOfLastStateChange" nillable="true" type="cim:cimDateTime">
3371     <xs:annotation>
3372         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3373             <cim:Qualifier name="Description" type="cim:cimString">The date or time when the
3374 EnabledState of the element last changed. If the state of the element has not changed and this
3375 property is populated, then it must be set to a 0 interval value. If a state change was
3376 requested, but rejected or not yet processed, the property must not be updated.</cim:Qualifier>
3377             <cim:Classorigin name="CIM_EnabledLogicalElement"/>
3378         </xs:appinfo>
3379     </xs:annotation>
3380 </xs:element>
3381 <xs:element name="AvailableRequestedStates" nillable="true">
3382     <xs:annotation>
3383         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">

```

```

3384     <cim:Qualifier name="Description" type="cim:cimString">AvailableRequestedStates indicates
3385 the possible values for the RequestedState parameter of the method RequestStateChange, used to
3386 initiate a state change. The values listed shall be a subset of the values contained in the
3387 RequestedStatesSupported property of the associated instance of
3388 CIM_EnabledLogicalElementCapabilities where the values selected are a function of the current
3389 state of the CIM_EnabledLogicalElement. This property may be non-null if an implementation is
3390 able to advertise the set of possible values as a function of the current state. This property
3391 shall be null if an implementation is unable to determine the set of possible values as a
3392 function of the current state.</cim:Qualifier>
3393     <cim:Qualifier name="ModelCorrespondence"
3394 type="cim:cimString">CIM_EnabledLogicalElement.RequestStateChange</cim:Qualifier>
3395     <cim:Qualifier name="ModelCorrespondence"
3396 type="cim:cimString">CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported</cim:Qualifie
3397 r>
3398     <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Enabled</cim:Qualifier>
3399     <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Disabled</cim:Qualifier>
3400     <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Shut Down</cim:Qualifier>
3401     <cim:Qualifier name="Values" type="cim:cimString" valuemap="6">Offline</cim:Qualifier>
3402     <cim:Qualifier name="Values" type="cim:cimString" valuemap="7">Test</cim:Qualifier>
3403     <cim:Qualifier name="Values" type="cim:cimString" valuemap="8">Defer</cim:Qualifier>
3404     <cim:Qualifier name="Values" type="cim:cimString" valuemap="9">Quiesce</cim:Qualifier>
3405     <cim:Qualifier name="Values" type="cim:cimString" valuemap="10">Reboot</cim:Qualifier>
3406     <cim:Qualifier name="Values" type="cim:cimString" valuemap="11">Reset</cim:Qualifier>
3407     <cim:Qualifier name="Values" type="cim:cimString" valuemap="..">DMTF
3408 Reserved</cim:Qualifier>
3409     <cim:Classorigin name="CIM_EnabledLogicalElement" />
3410 </xs:appinfo>
3411 </xs:annotation>
3412 <xs:complexType>
3413   <xs:simpleContent>
3414     <xs:restriction base="cim:cimAnySimpleType">
3415       <xs:simpleType>
3416         <xs:union>
3417           <xs:simpleType>
3418             <xs:restriction base="xs:unsignedShort">
3419               <xs:enumeration value="2" />
3420               <xs:enumeration value="3" />
3421               <xs:enumeration value="4" />
3422               <xs:enumeration value="6" />
3423               <xs:enumeration value="7" />
3424               <xs:enumeration value="8" />
3425               <xs:enumeration value="9" />
3426               <xs:enumeration value="10" />
3427               <xs:enumeration value="11" />
3428             </xs:restriction>
3429           </xs:simpleType>
3430           <xs:simpleType>
3431             <xs:union>
3432               <xs:simpleType>
3433                 <xs:restriction base="xs:unsignedShort">
3434                   <xs:maxInclusive value="1" />
3435                 </xs:restriction>
3436               </xs:simpleType>
3437               <xs:simpleType>
3438                 <xs:restriction base="xs:unsignedShort">
3439                   <xs:enumeration value="5" />

```



```

3440         </xs:restriction>
3441     </xs:simpleType>
3442     <xs:simpleType>
3443         <xs:restriction base="xs:unsignedShort">
3444             <xs:minInclusive value="12"/>
3445         </xs:restriction>
3446     </xs:simpleType>
3447 </xs:union>
3448 </xs:simpleType>
3449 </xs:union>
3450 </xs:simpleType>
3451 <xs:anyAttribute namespace="##any" processContents="lax"/>
3452 </xs:restriction>
3453 </xs:simpleContent>
3454 </xs:complexType>
3455 </xs:element>
3456 <xs:element name="TransitioningToState" nillable="true">
3457     <xs:annotation>
3458         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3459             <cim:Qualifier name="Description" type="cim:cimString">TransitioningToState indicates the
3460 target state to which the instance is transitioning.
3461 A value of 5 "No Change" shall indicate that no transition is in progress.A value of 12 "Not
3462 Applicable" shall indicate the implementation does not support representing ongoing transitions.
3463 A value other than 5 or 12 shall identify the state to which the element is in the process of
3464 transitioning.</cim:Qualifier>
3465         <cim:Qualifier name="ModelCorrespondence"
3466 type="cim:cimString">CIM_EnabledLogicalElement.RequestStateChange</cim:Qualifier>
3467         <cim:Qualifier name="ModelCorrespondence"
3468 type="cim:cimString">CIM_EnabledLogicalElement.RequestedState</cim:Qualifier>
3469         <cim:Qualifier name="ModelCorrespondence"
3470 type="cim:cimString">CIM_EnabledLogicalElement.EnabledState</cim:Qualifier>
3471         <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Unknown</cim:Qualifier>
3472         <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Enabled</cim:Qualifier>
3473         <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Disabled</cim:Qualifier>
3474         <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Shut Down</cim:Qualifier>
3475         <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">No Change</cim:Qualifier>
3476         <cim:Qualifier name="Values" type="cim:cimString" valuemap="6">Offline</cim:Qualifier>
3477         <cim:Qualifier name="Values" type="cim:cimString" valuemap="7">Test</cim:Qualifier>
3478         <cim:Qualifier name="Values" type="cim:cimString" valuemap="8">Defer</cim:Qualifier>
3479         <cim:Qualifier name="Values" type="cim:cimString" valuemap="9">Quiesce</cim:Qualifier>
3480         <cim:Qualifier name="Values" type="cim:cimString" valuemap="10">Reboot</cim:Qualifier>
3481         <cim:Qualifier name="Values" type="cim:cimString" valuemap="11">Reset</cim:Qualifier>
3482         <cim:Qualifier name="Values" type="cim:cimString" valuemap="12">Not
3483 Applicable</cim:Qualifier>
3484         <cim:Qualifier name="Values" type="cim:cimString" valuemap="..">DMTF
3485 Reserved</cim:Qualifier>
3486         <cim:Classorigin name="CIM_EnabledLogicalElement"/>
3487         <cim:Defaultvalue type="cim:cimUnsignedShort">12</cim:Defaultvalue>
3488     </xs:appinfo>
3489 </xs:annotation>
3490 <xs:complexType>
3491     <xs:simpleContent>
3492         <xs:restriction base="cim:cimAnySimpleType">
3493             <xs:simpleType>
3494                 <xs:union>

```

```

3495     <xs:simpleType>
3496         <xs:restriction base="xs:unsignedShort">
3497             <xs:enumeration value="0"/>
3498             <xs:enumeration value="2"/>
3499             <xs:enumeration value="3"/>
3500             <xs:enumeration value="4"/>
3501             <xs:enumeration value="5"/>
3502             <xs:enumeration value="6"/>
3503             <xs:enumeration value="7"/>
3504             <xs:enumeration value="8"/>
3505             <xs:enumeration value="9"/>
3506             <xs:enumeration value="10"/>
3507             <xs:enumeration value="11"/>
3508             <xs:enumeration value="12"/>
3509         </xs:restriction>
3510     </xs:simpleType>
3511     <xs:simpleType>
3512         <xs:union>
3513             <xs:simpleType>
3514                 <xs:restriction base="xs:unsignedShort">
3515                     <xs:enumeration value="1"/>
3516                 </xs:restriction>
3517             </xs:simpleType>
3518             <xs:simpleType>
3519                 <xs:restriction base="xs:unsignedShort">
3520                     <xs:minInclusive value="13"/>
3521                 </xs:restriction>
3522             </xs:simpleType>
3523         </xs:union>
3524     </xs:simpleType>
3525 </xs:union>
3526 </xs:simpleType>
3527     <xs:anyAttribute namespace="##any" processContents="lax"/>
3528 </xs:restriction>
3529 </xs:simpleContent>
3530 </xs:complexType>
3531 </xs:element>
3532 <xs:element name="InstallDate" nillable="true" type="cim:cimDateTime">
3533     <xs:annotation>
3534         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3535             <cim:Qualifier name="Description" type="cim:cimString">A datetime value that indicates
3536 when the object was installed. Lack of a value does not indicate that the object is not
3537 installed.</cim:Qualifier>
3538             <cim:Qualifier name="MappingStrings"
3539 type="cim:cimString">MIF.DMTF|ComponentID|001.5</cim:Qualifier>
3540             <cim:Classorigin name="CIM_ManagedSystemElement"/>
3541         </xs:appinfo>
3542     </xs:annotation>
3543 </xs:element>
3544 <xs:element name="OperationalStatus" nillable="true">
3545     <xs:annotation>
3546         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3547             <cim:Qualifier name="Description" type="cim:cimString">Indicates the current statuses of
3548 the element. Various operational statuses are defined. Many of the enumeration's values are self-
3549 explanatory. However, a few are not and are described here in more detail.

```

3550 "Stressed" indicates that the element is functioning, but needs attention. Examples of "Stressed"
3551 states are overload, overheated, and so on.

3552 "Predictive Failure" indicates that an element is functioning nominally but predicting a failure
3553 in the near future.

3554 "In Service" describes an element being configured, maintained, cleaned, or otherwise
3555 administered.

3556 "No Contact" indicates that the monitoring system has knowledge of this element, but has never
3557 been able to establish communications with it.

3558 "Lost Communication" indicates that the ManagedSystem Element is known to exist and has been
3559 contacted successfully in the past, but is currently unreachable.

3560 "Stopped" and "Aborted" are similar, although the former implies a clean and orderly stop, while
3561 the latter implies an abrupt stop where the state and configuration of the element might need to
3562 be updated.

3563 "Dormant" indicates that the element is inactive or quiesced.

3564 "Supporting Entity in Error" indicates that this element might be "OK" but that another element,
3565 on which it is dependent, is in error. An example is a network service or endpoint that cannot
3566 function due to lower-layer networking problems.

3567 "Completed" indicates that the element has completed its operation. This value should be combined
3568 with either OK, Error, or Degraded so that a client can tell if the complete operation Completed
3569 with OK (passed), Completed with Error (failed), or Completed with Degraded (the operation
3570 finished, but it did not complete OK or did not report an error).

3571 "Power Mode" indicates that the element has additional power model information contained in the
3572 Associated PowerManagementService association.

3573 "Relocating" indicates the element is being relocated.

3574 OperationalStatus replaces the Status property on ManagedSystemElement to provide a consistent
3575 approach to enumerations, to address implementation needs for an array property, and to provide a
3576 migration path from today's environment to the future. This change was not made earlier because
3577 it required the deprecated qualifier. Due to the widespread use of the existing Status property
3578 in management applications, it is strongly recommended that providers or instrumentation provide
3579 both the Status and OperationalStatus properties. Further, the first value of OperationalStatus
3580 should contain the primary status for the element. When instrumented, Status (because it is
3581 single-valued) should also provide the primary status of the element.</cim:Qualifier>

```

3582     <cim:Qualifier name="ArrayType" type="cim:cimString">Indexed</cim:Qualifier>
3583     <cim:Qualifier name="ModelCorrespondence"
3584 type="cim:cimString">CIM_ManagedSystemElement.StatusDescriptions</cim:Qualifier>
3585     <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Unknown</cim:Qualifier>
3586     <cim:Qualifier name="Values" type="cim:cimString" valuemap="1">Other</cim:Qualifier>
3587     <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">OK</cim:Qualifier>
3588     <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Degraded</cim:Qualifier>
3589     <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Stressed</cim:Qualifier>
3590     <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">Predictive
3591 Failure</cim:Qualifier>
3592     <cim:Qualifier name="Values" type="cim:cimString" valuemap="6">Error</cim:Qualifier>
3593     <cim:Qualifier name="Values" type="cim:cimString" valuemap="7">Non-Recoverable
3594 Error</cim:Qualifier>
3595     <cim:Qualifier name="Values" type="cim:cimString" valuemap="8">Starting</cim:Qualifier>
3596     <cim:Qualifier name="Values" type="cim:cimString" valuemap="9">Stopping</cim:Qualifier>
3597     <cim:Qualifier name="Values" type="cim:cimString" valuemap="10">Stopped</cim:Qualifier>
3598     <cim:Qualifier name="Values" type="cim:cimString" valuemap="11">In
3599 Service</cim:Qualifier>
3600     <cim:Qualifier name="Values" type="cim:cimString" valuemap="12">No
3601 Contact</cim:Qualifier>
3602     <cim:Qualifier name="Values" type="cim:cimString" valuemap="13">Lost
3603 Communication</cim:Qualifier>
3604     <cim:Qualifier name="Values" type="cim:cimString" valuemap="14">Aborted</cim:Qualifier>
3605     <cim:Qualifier name="Values" type="cim:cimString" valuemap="15">Dormant</cim:Qualifier>
3606     <cim:Qualifier name="Values" type="cim:cimString" valuemap="16">Supporting Entity in
3607 Error</cim:Qualifier>
3608     <cim:Qualifier name="Values" type="cim:cimString" valuemap="17">Completed</cim:Qualifier>

```

3609

```
<cim:Qualifier name="Values" type="cim:cimString" valuemap="18">Power
```

```

3663         </xs:simpleType>
3664     </xs:union>
3665 </xs:simpleType>
3666     <xs:anyAttribute namespace="##any" processContents="lax"/>
3667 </xs:restriction>
3668 </xs:simpleContent>
3669 </xs:complexType>
3670 </xs:element>
3671 <xs:element name="StatusDescriptions" nillable="true" type="cim:cimString">
3672     <xs:annotation>
3673         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3674             <cim:Qualifier name="Description" type="cim:cimString">Strings describing the various
3675 OperationalStatus array values. For example, if "Stopping" is the value assigned to
3676 OperationalStatus, then this property may contain an explanation as to why an object is being
3677 stopped. Note that entries in this array are correlated with those at the same array index in
3678 OperationalStatus.</cim:Qualifier>
3679             <cim:Qualifier name="ArrayType" type="cim:cimString">Indexed</cim:Qualifier>
3680             <cim:Qualifier name="ModelCorrespondence"
3681 type="cim:cimString">CIM_ManagedSystemElement.OperationalStatus</cim:Qualifier>
3682             <cim:Classorigin name="CIM_ManagedSystemElement" />
3683         </xs:appinfo>
3684     </xs:annotation>
3685 </xs:element>
3686 <xs:element name="Status" nillable="true">
3687     <xs:annotation>
3688         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3689             <cim:Qualifier name="Deprecated"
3690 type="cim:cimString">CIM_ManagedSystemElement.OperationalStatus</cim:Qualifier>
3691             <cim:Qualifier name="Description" type="cim:cimString">A string indicating the current
3692 status of the object. Various operational and non-operational statuses are defined. This property
3693 is deprecated in lieu of OperationalStatus, which includes the same semantics in its enumeration.
3694 This change is made for 3 reasons:
3695 1) Status is more correctly defined as an array. This definition overcomes the limitation of
3696 describing status using a single value, when it is really a multi-valued property (for example,
3697 an element might be OK AND Stopped.
3698 2) A MaxLen of 10 is too restrictive and leads to unclear enumerated values.
3699 3) The change to a uint16 data type was discussed when CIM V2.0 was defined. However, existing
3700 V1.0 implementations used the string property and did not want to modify their code. Therefore,
3701 Status was grandfathered into the Schema. Use of the deprecated qualifier allows the maintenance
3702 of the existing property, but also permits an improved definition using
3703 OperationalStatus.</cim:Qualifier>
3704             <cim:Qualifier name="MaxLen" type="cim:cimUnsignedInt">10</cim:Qualifier>
3705             <cim:Qualifier name="Values" type="cim:cimString" valuemap="OK" xsi:nil="true"/>
3706             <cim:Qualifier name="Values" type="cim:cimString" valuemap="Error" xsi:nil="true"/>
3707             <cim:Qualifier name="Values" type="cim:cimString" valuemap="Degraded" xsi:nil="true"/>
3708             <cim:Qualifier name="Values" type="cim:cimString" valuemap="Unknown" xsi:nil="true"/>
3709             <cim:Qualifier name="Values" type="cim:cimString" valuemap="Pred Fail" xsi:nil="true"/>
3710             <cim:Qualifier name="Values" type="cim:cimString" valuemap="Starting" xsi:nil="true"/>
3711             <cim:Qualifier name="Values" type="cim:cimString" valuemap="Stopping" xsi:nil="true"/>
3712             <cim:Qualifier name="Values" type="cim:cimString" valuemap="Service" xsi:nil="true"/>
3713             <cim:Qualifier name="Values" type="cim:cimString" valuemap="Stressed" xsi:nil="true"/>
3714             <cim:Qualifier name="Values" type="cim:cimString" valuemap="NonRecover" xsi:nil="true"/>
3715             <cim:Qualifier name="Values" type="cim:cimString" valuemap="No Contact" xsi:nil="true"/>
3716             <cim:Qualifier name="Values" type="cim:cimString" valuemap="Lost Comm" xsi:nil="true"/>
3717             <cim:Qualifier name="Values" type="cim:cimString" valuemap="Stopped" xsi:nil="true"/>
3718             <cim:Classorigin name="CIM_ManagedSystemElement" />
3719         </xs:appinfo>

```

```

3720 </xs:annotation>
3721 <xs:complexType>
3722   <xs:simpleContent>
3723     <xs:restriction base="cim:cimString">
3724       <xs:maxLength value="10"/>
3725       <xs:enumeration value="OK"/>
3726       <xs:enumeration value="Error"/>
3727       <xs:enumeration value="Degraded"/>
3728       <xs:enumeration value="Unknown"/>
3729       <xs:enumeration value="Pred Fail"/>
3730       <xs:enumeration value="Starting"/>
3731       <xs:enumeration value="Stopping"/>
3732       <xs:enumeration value="Service"/>
3733       <xs:enumeration value="Stressed"/>
3734       <xs:enumeration value="NonRecover"/>
3735       <xs:enumeration value="No Contact"/>
3736       <xs:enumeration value="Lost Comm"/>
3737       <xs:enumeration value="Stopped"/>
3738       <xs:anyAttribute namespace="##any" processContents="lax"/>
3739     </xs:restriction>
3740   </xs:simpleContent>
3741 </xs:complexType>
3742 </xs:element>
3743 <xs:element name="HealthState" nillable="true">
3744   <xs:annotation>
3745     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3746       <cim:Qualifier name="Description" type="cim:cimString">Indicates the current health of
3747 the element. This attribute expresses the health of this element but not necessarily that of its
3748 subcomponents. The possible values are 0 to 30, where 5 means the element is entirely healthy and
3749 30 means the element is completely non-functional. The following continuum is defined:
3750 "Non-recoverable Error" (30) - The element has completely failed, and recovery is not possible.
3751 All functionality provided by this element has been lost.
3752 "Critical Failure" (25) - The element is non-functional and recovery might not be possible.
3753 "Major Failure" (20) - The element is failing. It is possible that some or all of the
3754 functionality of this component is degraded or not working.
3755 "Minor Failure" (15) - All functionality is available but some might be degraded.
3756 "Degraded/Warning" (10) - The element is in working order and all functionality is provided.
3757 However, the element is not working to the best of its abilities. For example, the element might
3758 not be operating at optimal performance or it might be reporting recoverable errors.
3759 "OK" (5) - The element is fully functional and is operating within normal operational parameters
3760 and without error.
3761 "Unknown" (0) - The implementation cannot report on HealthState at this time.
3762 DMTF has reserved the unused portion of the continuum for additional HealthStates in the
3763 future.</cim:Qualifier>
3764     <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Unknown</cim:Qualifier>
3765     <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">OK</cim:Qualifier>
3766     <cim:Qualifier name="Values" type="cim:cimString"
3767 valuemap="10">Degraded/Warning</cim:Qualifier>
3768     <cim:Qualifier name="Values" type="cim:cimString" valuemap="15">Minor
3769 failure</cim:Qualifier>
3770     <cim:Qualifier name="Values" type="cim:cimString" valuemap="20">Major
3771 failure</cim:Qualifier>
3772     <cim:Qualifier name="Values" type="cim:cimString" valuemap="25">Critical
3773 failure</cim:Qualifier>
3774     <cim:Qualifier name="Values" type="cim:cimString" valuemap="30">Non-recoverable
3775 error</cim:Qualifier>

```

```

3776     <cim:Qualifier name="Values" type="cim:cimString" valuemap="..">DMTF
3777 Reserved</cim:Qualifier>
3778     <cim:Classorigin name="CIM_ManagedSystemElement" />
3779     </xs:appinfo>
3780 </xs:annotation>
3781 <xs:complexType>
3782   <xs:simpleContent>
3783     <xs:restriction base="cim:cimAnySimpleType">
3784       <xs:simpleType>
3785         <xs:union>
3786           <xs:simpleType>
3787             <xs:restriction base="xs:unsignedShort">
3788               <xs:enumeration value="0" />
3789               <xs:enumeration value="5" />
3790               <xs:enumeration value="10" />
3791               <xs:enumeration value="15" />
3792               <xs:enumeration value="20" />
3793               <xs:enumeration value="25" />
3794               <xs:enumeration value="30" />
3795             </xs:restriction>
3796           </xs:simpleType>
3797           <xs:simpleType>
3798             <xs:union>
3799               <xs:simpleType>
3800                 <xs:restriction base="xs:unsignedShort">
3801                   <xs:minInclusive value="1" />
3802                   <xs:maxInclusive value="4" />
3803                 </xs:restriction>
3804               </xs:simpleType>
3805               <xs:simpleType>
3806                 <xs:restriction base="xs:unsignedShort">
3807                   <xs:minInclusive value="6" />
3808                   <xs:maxInclusive value="9" />
3809                 </xs:restriction>
3810               </xs:simpleType>
3811               <xs:simpleType>
3812                 <xs:restriction base="xs:unsignedShort">
3813                   <xs:minInclusive value="11" />
3814                   <xs:maxInclusive value="14" />
3815                 </xs:restriction>
3816               </xs:simpleType>
3817               <xs:simpleType>
3818                 <xs:restriction base="xs:unsignedShort">
3819                   <xs:minInclusive value="16" />
3820                   <xs:maxInclusive value="19" />
3821                 </xs:restriction>
3822               </xs:simpleType>
3823               <xs:simpleType>
3824                 <xs:restriction base="xs:unsignedShort">
3825                   <xs:minInclusive value="21" />
3826                   <xs:maxInclusive value="24" />
3827                 </xs:restriction>
3828               </xs:simpleType>
3829             </xs:union>

```

```

3830         <xs:restriction base="xs:unsignedShort">
3831             <xs:minInclusive value="26"/>
3832             <xs:maxInclusive value="29"/>
3833         </xs:restriction>
3834     </xs:simpleType>
3835     <xs:simpleType>
3836         <xs:restriction base="xs:unsignedShort">
3837             <xs:minInclusive value="31"/>
3838         </xs:restriction>
3839     </xs:simpleType>
3840 </xs:union>
3841 </xs:simpleType>
3842 </xs:union>
3843 </xs:simpleType>
3844     <xs:anyAttribute namespace="##any" processContents="lax"/>
3845 </xs:restriction>
3846 </xs:simpleContent>
3847 </xs:complexType>
3848 </xs:element>
3849 <xs:element name="CommunicationStatus" nillable="true">
3850     <xs:annotation>
3851         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3852             <cim:Qualifier name="Description" type="cim:cimString">CommunicationStatus indicates the
3853 ability of the instrumentation to communicate with the underlying ManagedElement.
3854 CommunicationStatus consists of one of the following values: Unknown, None, Communication OK,
3855 Lost Communication, or No Contact.
3856 A Null return indicates the implementation (provider) does not implement this property.
3857 "Unknown" indicates the implementation is in general capable of returning this property, but is
3858 unable to do so at this time.
3859 "Not Available" indicates that the implementation (provider) is capable of returning a value for
3860 this property, but not ever for this particular piece of hardware/software or the property is
3861 intentionally not used because it adds no meaningful information (as in the case of a property
3862 that is intended to add additional info to another property).
3863 "Communication OK " indicates communication is established with the element, but does not convey
3864 any quality of service.
3865 "No Contact" indicates that the monitoring system has knowledge of this element, but has never
3866 been able to establish communications with it.
3867 "Lost Communication" indicates that the Managed Element is known to exist and has been contacted
3868 successfully in the past, but is currently unreachable.</cim:Qualifier>
3869         <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Unknown</cim:Qualifier>
3870         <cim:Qualifier name="Values" type="cim:cimString" valuemap="1">Not
3871 Available</cim:Qualifier>
3872         <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Communication
3873 OK</cim:Qualifier>
3874         <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Lost
3875 Communication</cim:Qualifier>
3876         <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">No Contact</cim:Qualifier>
3877         <cim:Qualifier name="Values" type="cim:cimString" valuemap="..">DMTF
3878 Reserved</cim:Qualifier>
3879         <cim:Qualifier name="Values" type="cim:cimString" valuemap="0x8000..">Vendor
3880 Reserved</cim:Qualifier>
3881         <cim:Classorigin name="CIM_ManagedSystemElement"/>
3882     </xs:appinfo>
3883 </xs:annotation>
3884 </xs:complexType>
3885 <xs:simpleContent>
3886     <xs:restriction base="cim:cimAnySimpleType">

```



```

3887     <xs:simpleType>
3888         <xs:union>
3889             <xs:simpleType>
3890                 <xs:restriction base="xs:unsignedShort">
3891                     <xs:enumeration value="0"/>
3892                     <xs:enumeration value="1"/>
3893                     <xs:enumeration value="2"/>
3894                     <xs:enumeration value="3"/>
3895                     <xs:enumeration value="4"/>
3896                 </xs:restriction>
3897             </xs:simpleType>
3898             <xs:simpleType>
3899                 <xs:union>
3900                     <xs:simpleType>
3901                         <xs:restriction base="xs:unsignedShort">
3902                             <xs:minInclusive value="5"/>
3903                             <xs:maxInclusive value="32767"/>
3904                         </xs:restriction>
3905                     </xs:simpleType>
3906                 </xs:union>
3907             </xs:simpleType>
3908             <xs:simpleType>
3909                 <xs:restriction base="xs:unsignedShort">
3910                     <xs:minInclusive value="32768"/>
3911                 </xs:restriction>
3912             </xs:simpleType>
3913         </xs:union>
3914     </xs:simpleType>
3915     <xs:anyAttribute namespace="##any" processContents="lax"/>
3916 </xs:restriction>
3917 </xs:simpleContent>
3918 </xs:complexType>
3919 </xs:element>
3920 <xs:element name="DetailedStatus" nillable="true">
3921     <xs:annotation>
3922         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
3923             <cim:Qualifier name="Description" type="cim:cimString">DetailedStatus compliments
3924 PrimaryStatus with additional status detail. It consists of one of the following values: Not
3925 Available, No Additional Information, Stressed, Predictive Failure, Error, Non-Recoverable Error,
3926 SupportingEntityInError. Detailed status is used to expand upon the PrimaryStatus of the element.
3927 A Null return indicates the implementation (provider) does not implement this property.
3928 "Not Available" indicates that the implementation (provider) is capable of returning a value for
3929 this property, but not ever for this particular piece of hardware/software or the property is
3930 intentionally not used because it adds no meaningful information (as in the case of a property
3931 that is intended to add additional info to another property).
3932 "No Additional Information" indicates that the element is functioning normally as indicated by
3933 PrimaryStatus = "OK".
3934 "Stressed" indicates that the element is functioning, but needs attention. Examples of "Stressed"
3935 states are overload, overheated, and so on.
3936 "Predictive Failure" indicates that an element is functioning normally but a failure is predicted
3937 in the near future.
3938 "Non-Recoverable Error " indicates that this element is in an error condition that requires human
3939 intervention.
3940 "Supporting Entity in Error" indicates that this element might be "OK" but that another element,
3941 on which it is dependent, is in error. An example is a network service or endpoint that cannot
3942 function due to lower-layer networking problems.</cim:Qualifier>

```

```

3943     <cim:Qualifier name="ModelCorrespondence"
3944 type="cim:cimString">CIM_EnabledLogicalElement.PrimaryStatus</cim:Qualifier>
3945     <cim:Qualifier name="ModelCorrespondence"
3946 type="cim:cimString">CIM_ManagedSystemElement.HealthState</cim:Qualifier>
3947     <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Not
3948 Available</cim:Qualifier>
3949     <cim:Qualifier name="Values" type="cim:cimString" valuemap="1">No Additional
3950 Information</cim:Qualifier>
3951     <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Stressed</cim:Qualifier>
3952     <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Predictive
3953 Failure</cim:Qualifier>
3954     <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Non-Recoverable
3955 Error</cim:Qualifier>
3956     <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">Supporting Entity in
3957 Error</cim:Qualifier>
3958     <cim:Qualifier name="Values" type="cim:cimString" valuemap="..">DMTF
3959 Reserved</cim:Qualifier>
3960     <cim:Qualifier name="Values" type="cim:cimString" valuemap="0x8000..">Vendor
3961 Reserved</cim:Qualifier>
3962     <cim:Classorigin name="CIM_ManagedSystemElement" />
3963     </xs:appinfo>
3964     </xs:annotation>
3965     <xs:complexType>
3966     <xs:simpleContent>
3967     <xs:restriction base="cim:cimAnySimpleType">
3968     <xs:simpleType>
3969     <xs:union>
3970     <xs:simpleType>
3971     <xs:restriction base="xs:unsignedShort">
3972     <xs:enumeration value="0"/>
3973     <xs:enumeration value="1"/>
3974     <xs:enumeration value="2"/>
3975     <xs:enumeration value="3"/>
3976     <xs:enumeration value="4"/>
3977     <xs:enumeration value="5"/>
3978     </xs:restriction>
3979     </xs:simpleType>
3980     <xs:simpleType>
3981     <xs:union>
3982     <xs:simpleType>
3983     <xs:restriction base="xs:unsignedShort">
3984     <xs:minInclusive value="6"/>
3985     <xs:maxInclusive value="32767"/>
3986     </xs:restriction>
3987     </xs:simpleType>
3988     </xs:union>
3989     </xs:simpleType>
3990     <xs:simpleType>
3991     <xs:restriction base="xs:unsignedShort">
3992     <xs:minInclusive value="32768"/>
3993     </xs:restriction>
3994     </xs:simpleType>
3995     </xs:union>
3996     </xs:simpleType>
3997     <xs:anyAttribute namespace="##any" processContents="lax"/>
3998     </xs:restriction>

```

```

3999     </xs:simpleContent>
4000     </xs:complexType>
4001 </xs:element>
4002 <xs:element name="OperatingStatus" nillable="true">
4003     <xs:annotation>
4004         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
4005             <cim:Qualifier name="Description" type="cim:cimString">OperatingStatus provides a current
4006 status value for the operational condition of the element and can be used for providing more
4007 detail with respect to the value of EnabledState. It can also provide the transitional states
4008 when an element is transitioning from one state to another, such as when an element is
4009 transitioning between EnabledState and RequestedState, as well as other transitional conditions.
4010 OperatingStatus consists of one of the following values: Unknown, Not Available, In Service,
4011 Starting, Stopping, Stopped, Aborted, Dormant, Completed, Migrating, Emmigrating, Immigrating,
4012 Snapshotting. Shutting Down, In Test
4013 A Null return indicates the implementation (provider) does not implement this property.
4014 "Unknown" indicates the implementation is in general capable of returning this property, but is
4015 unable to do so at this time.
4016 "None" indicates that the implementation (provider) is capable of returning a value for this
4017 property, but not ever for this particular piece of hardware/software or the property is
4018 intentionally not used because it adds no meaningful information (as in the case of a property
4019 that is intended to add additional info to another property).
4020 "Servicing" describes an element being configured, maintained, cleaned, or otherwise
4021 administered.
4022 "Starting" describes an element being initialized.
4023 "Stopping" describes an element being brought to an orderly stop.
4024 "Stopped" and "Aborted" are similar, although the former implies a clean and orderly stop, while
4025 the latter implies an abrupt stop where the state and configuration of the element might need to
4026 be updated.
4027 "Dormant" indicates that the element is inactive or quiesced.
4028 "Completed" indicates that the element has completed its operation. This value should be combined
4029 with either OK, Error, or Degraded in the PrimaryStatus so that a client can tell if the complete
4030 operation Completed with OK (passed), Completed with Error (failed), or Completed with Degraded
4031 (the operation finished, but it did not complete OK or did not report an error).
4032 "Migrating" element is being moved between host elements.
4033 "Immigrating" element is being moved to new host element.
4034 "Emigrating" element is being moved away from host element.
4035 "Shutting Down" describes an element being brought to an abrupt stop.
4036 "In Test" element is performing test functions.
4037 "Transitioning" describes an element that is between states, that is, it is not fully available
4038 in either its previous state or its next state. This value should be used if other values
4039 indicating a transition to a specific state are not applicable.
4040 "In Service" describes an element that is in service and operational.</cim:Qualifier>
4041     <cim:Qualifier name="ModelCorrespondence"
4042 type="cim:cimString">CIM_EnabledLogicalElement.EnabledState</cim:Qualifier>
4043     <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Unknown</cim:Qualifier>
4044     <cim:Qualifier name="Values" type="cim:cimString" valuemap="1">Not
4045 Available</cim:Qualifier>
4046     <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Servicing</cim:Qualifier>
4047     <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Starting</cim:Qualifier>
4048     <cim:Qualifier name="Values" type="cim:cimString" valuemap="4">Stopping</cim:Qualifier>
4049     <cim:Qualifier name="Values" type="cim:cimString" valuemap="5">Stopped</cim:Qualifier>
4050     <cim:Qualifier name="Values" type="cim:cimString" valuemap="6">Aborted</cim:Qualifier>
4051     <cim:Qualifier name="Values" type="cim:cimString" valuemap="7">Dormant</cim:Qualifier>
4052     <cim:Qualifier name="Values" type="cim:cimString" valuemap="8">Completed</cim:Qualifier>
4053     <cim:Qualifier name="Values" type="cim:cimString" valuemap="9">Migrating</cim:Qualifier>
4054     <cim:Qualifier name="Values" type="cim:cimString"
4055 valuemap="10">Emigrating</cim:Qualifier>
4056     <cim:Qualifier name="Values" type="cim:cimString"
4057 valuemap="11">Immigrating</cim:Qualifier>

```

```

4058     <cim:Qualifier name="Values" type="cim:cimString"
4059 valuemap="12">Snapshotting</cim:Qualifier>
4060     <cim:Qualifier name="Values" type="cim:cimString" valuemap="13">Shutting
4061 Down</cim:Qualifier>
4062     <cim:Qualifier name="Values" type="cim:cimString" valuemap="14">In Test</cim:Qualifier>
4063     <cim:Qualifier name="Values" type="cim:cimString"
4064 valuemap="15">Transitioning</cim:Qualifier>
4065     <cim:Qualifier name="Values" type="cim:cimString" valuemap="16">In
4066 Service</cim:Qualifier>
4067     <cim:Qualifier name="Values" type="cim:cimString" valuemap="..">DMTF
4068 Reserved</cim:Qualifier>
4069     <cim:Qualifier name="Values" type="cim:cimString" valuemap="0x8000..">Vendor
4070 Reserved</cim:Qualifier>
4071     <cim:Classorigin name="CIM_ManagedSystemElement"/>
4072   </xs:appinfo>
4073 </xs:annotation>
4074 <xs:complexType>
4075   <xs:simpleContent>
4076     <xs:restriction base="cim:cimAnySimpleType">
4077       <xs:simpleType>
4078         <xs:union>
4079           <xs:simpleType>
4080             <xs:restriction base="xs:unsignedShort">
4081               <xs:enumeration value="0"/>
4082               <xs:enumeration value="1"/>
4083               <xs:enumeration value="2"/>
4084               <xs:enumeration value="3"/>
4085               <xs:enumeration value="4"/>
4086               <xs:enumeration value="5"/>
4087               <xs:enumeration value="6"/>
4088               <xs:enumeration value="7"/>
4089               <xs:enumeration value="8"/>
4090               <xs:enumeration value="9"/>
4091               <xs:enumeration value="10"/>
4092               <xs:enumeration value="11"/>
4093               <xs:enumeration value="12"/>
4094               <xs:enumeration value="13"/>
4095               <xs:enumeration value="14"/>
4096               <xs:enumeration value="15"/>
4097               <xs:enumeration value="16"/>
4098             </xs:restriction>
4099           </xs:simpleType>
4100           <xs:simpleType>
4101             <xs:union>
4102               <xs:simpleType>
4103                 <xs:restriction base="xs:unsignedShort">
4104                   <xs:minInclusive value="17"/>
4105                   <xs:maxInclusive value="32767"/>
4106                 </xs:restriction>
4107               </xs:simpleType>
4108             </xs:union>
4109           </xs:simpleType>
4110           <xs:simpleType>
4111             <xs:restriction base="xs:unsignedShort">
4112               <xs:minInclusive value="32768"/>

```

```

4113         </xs:restriction>
4114     </xs:simpleType>
4115 </xs:union>
4116 </xs:simpleType>
4117     <xs:anyAttribute namespace="##any" processContents="lax"/>
4118 </xs:restriction>
4119 </xs:simpleContent>
4120 </xs:complexType>
4121 </xs:element>
4122 <xs:element name="PrimaryStatus" nillable="true">
4123     <xs:annotation>
4124         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
4125             <cim:Qualifier name="Description" type="cim:cimString">PrimaryStatus provides a high
4126 level status value, intended to align with Red-Yellow-Green type representation of status. It
4127 should be used in conjunction with DetailedStatus to provide high level and detailed health
4128 status of the ManagedElement and its subcomponents.
4129 PrimaryStatus consists of one of the following values: Unknown, OK, Degraded or Error. "Unknown"
4130 indicates the implementation is in general capable of returning this property, but is unable to
4131 do so at this time.
4132 "OK" indicates the ManagedElement is functioning normally.
4133 "Degraded" indicates the ManagedElement is functioning below normal.
4134 "Error" indicates the ManagedElement is in an Error condition.</cim:Qualifier>
4135         <cim:Qualifier name="ModelCorrespondence"
4136 type="cim:cimString">CIM_ManagedSystemElement.DetailedStatus</cim:Qualifier>
4137         <cim:Qualifier name="ModelCorrespondence"
4138 type="cim:cimString">CIM_ManagedSystemElement.HealthState</cim:Qualifier>
4139         <cim:Qualifier name="Values" type="cim:cimString" valuemap="0">Unknown</cim:Qualifier>
4140         <cim:Qualifier name="Values" type="cim:cimString" valuemap="1">OK</cim:Qualifier>
4141         <cim:Qualifier name="Values" type="cim:cimString" valuemap="2">Degraded</cim:Qualifier>
4142         <cim:Qualifier name="Values" type="cim:cimString" valuemap="3">Error</cim:Qualifier>
4143         <cim:Qualifier name="Values" type="cim:cimString" valuemap="..">DMTF
4144 Reserved</cim:Qualifier>
4145         <cim:Qualifier name="Values" type="cim:cimString" valuemap="0x8000..">Vendor
4146 Reserved</cim:Qualifier>
4147         <cim:Classorigin name="CIM_ManagedSystemElement"/>
4148     </xs:appinfo>
4149 </xs:annotation>
4150 <xs:complexType>
4151     <xs:simpleContent>
4152         <xs:restriction base="cim:cimAnySimpleType">
4153             <xs:simpleType>
4154                 <xs:union>
4155                     <xs:simpleType>
4156                         <xs:restriction base="xs:unsignedShort">
4157                             <xs:enumeration value="0"/>
4158                             <xs:enumeration value="1"/>
4159                             <xs:enumeration value="2"/>
4160                             <xs:enumeration value="3"/>
4161                         </xs:restriction>
4162                     </xs:simpleType>
4163                     <xs:simpleType>
4164                         <xs:union>
4165                             <xs:simpleType>
4166                                 <xs:restriction base="xs:unsignedShort">
4167                                     <xs:minInclusive value="4"/>
4168                                     <xs:maxInclusive value="32767"/>

```

```

4169         </xs:restriction>
4170     </xs:simpleType>
4171 </xs:union>
4172 </xs:simpleType>
4173 <xs:simpleType>
4174     <xs:restriction base="xs:unsignedShort">
4175         <xs:minInclusive value="32768"/>
4176     </xs:restriction>
4177 </xs:simpleType>
4178 </xs:union>
4179 </xs:simpleType>
4180 <xs:anyAttribute namespace="##any" processContents="lax"/>
4181 </xs:restriction>
4182 </xs:simpleContent>
4183 </xs:complexType>
4184 </xs:element>
4185 <xs:element name="InstanceID" nillable="true" type="cim:cimString">
4186     <xs:annotation>
4187         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
4188             <cim:Qualifier name="Description" type="cim:cimString">InstanceID is an optional property
4189 that may be used to opaquely and uniquely identify an instance of this class within the scope of
4190 the instantiating Namespace. Various subclasses of this class may override this property to make
4191 it required, or a key. Such subclasses may also modify the preferred algorithms for ensuring
4192 uniqueness that are defined below.
4193 To ensure uniqueness within the NameSpace, the value of InstanceID should be constructed using
4194 the following "preferred" algorithm:
4195 &lt;OrgID&gt;:&lt;LocalID&gt;
4196 Where &lt;OrgID&gt; and &lt;LocalID&gt; are separated by a colon (:), and where &lt;OrgID&gt;
4197 must include a copyrighted, trademarked, or otherwise unique name that is owned by the business
4198 entity that is creating or defining the InstanceID or that is a registered ID assigned to the
4199 business entity by a recognized global authority. (This requirement is similar to the &lt;Schema
4200 Name&gt;_&lt;Class Name&gt; structure of Schema class names.) In addition, to ensure uniqueness,
4201 &lt;OrgID&gt; must not contain a colon (:). When using this algorithm, the first colon to appear
4202 in InstanceID must appear between &lt;OrgID&gt; and &lt;LocalID&gt;.
4203 &lt;LocalID&gt; is chosen by the business entity and should not be reused to identify different
4204 underlying (real-world) elements. If not null and the above "preferred" algorithm is not used,
4205 the defining entity must assure that the resulting InstanceID is not reused across any
4206 InstanceIDs produced by this or other providers for the NameSpace of this instance.
4207 If not set to null for DMTF-defined instances, the "preferred" algorithm must be used with the
4208 &lt;OrgID&gt; set to CIM.</cim:Qualifier>
4209         <cim:Classorigin name="CIM_ManagedElement"/>
4210     </xs:appinfo>
4211 </xs:annotation>
4212 </xs:element>
4213 <xs:element name="Caption" nillable="true">
4214     <xs:annotation>
4215         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
4216             <cim:Qualifier name="Description" type="cim:cimString">The Caption property is a short
4217 textual description (one- line string) of the object.</cim:Qualifier>
4218             <cim:Qualifier name="MaxLen" type="cim:cimUnsignedInt">64</cim:Qualifier>
4219             <cim:Classorigin name="CIM_ManagedElement"/>
4220         </xs:appinfo>
4221     </xs:annotation>
4222 </xs:complexType>
4223 <xs:simpleContent>
4224     <xs:restriction base="cim:cimString">
4225         <xs:maxLength value="64"/>

```

```

4226     <xs:anyAttribute namespace="##any" processContents="lax"/>
4227     </xs:restriction>
4228     </xs:simpleContent>
4229   </xs:complexType>
4230 </xs:element>
4231 <xs:element name="Description" nillable="true" type="cim:cimString">
4232   <xs:annotation>
4233     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
4234       <cim:Qualifier name="Description" type="cim:cimString">The Description property provides
4235 a textual description of the object.</cim:Qualifier>
4236       <cim:Classorigin name="CIM_ManagedElement"/>
4237     </xs:appinfo>
4238   </xs:annotation>
4239 </xs:element>
4240 <xs:element name="ElementName" nillable="true" type="cim:cimString">
4241   <xs:annotation>
4242     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
4243       <cim:Qualifier name="Description" type="cim:cimString">A user-friendly name for the
4244 object. This property allows each instance to define a user-friendly name in addition to its key
4245 properties, identity data, and description information.
4246 Note that the Name property of ManagedSystemElement is also defined as a user-friendly name. But,
4247 it is often subclassed to be a Key. It is not reasonable that the same property can convey both
4248 identity and a user-friendly name, without inconsistencies. Where Name exists and is not a Key
4249 (such as for instances of LogicalDevice), the same information can be present in both the Name
4250 and ElementName properties. Note that if there is an associated instance of
4251 CIM_EnabledLogicalElementCapabilities, restrictions on this properties may exist as defined in
4252 ElementNameMask and MaxElementNameLen properties defined in that class.</cim:Qualifier>
4253       <cim:Classorigin name="CIM_ManagedElement"/>
4254     </xs:appinfo>
4255   </xs:annotation>
4256 </xs:element>
4257 <xs:element name="Generation" nillable="true" type="cim:cimUnsignedLong">
4258   <xs:annotation>
4259     <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
4260       <cim:Qualifier name="Experimental" type="cim:cimBoolean">true</cim:Qualifier>
4261       <cim:Qualifier name="Description" type="cim:cimString">Generation is an optional,
4262 monotonically increasing property that may be used to identify a particular generation of the
4263 resource represented by this class.
4264 If Generation is supported by the implementation, its value shall not be null.
4265 Except as otherwise specified, a value (including null) of Generation specified at creation time
4266 shall be replaced by null if Generation is not supported by the implementation or shall be a,
4267 (possibly different), non-null value if the implementation does support Generation.
4268 After creation and if supported, Generation shall be updated, at least once per access, whenever
4269 the represented resource is modified, regardless of the source of the modification.
4270 Note: the Generation value only needs to be updated once between references, even if the resource
4271 is updated many times. The key point is to assure that it will be different if there have been
4272 updates, not to count each update.
4273 Note: unless otherwise specified, the value of Generation within one instance is not required to
4274 be coordinated with the value of Generation in any other instance.
4275 Note:the semantics of the instance, (as defined by its creation class), define the underlying
4276 resource. That underlying resource may be a collection or aggregation of resources. And, in that
4277 case, the semantics of the instance further define when updates to constituent resources also
4278 require updates to the Generation of the collective resource. Default behavior of composite
4279 aggregations should be to update the Generation of the composite whenever the Generation of a
4280 component is updated.
4281 Subclasses may define additional requirements for updates on some or all of related instances.
4282 For a particular instance, the value of Generation may wrap through zero, but the elapsed time
4283 between wraps shall be greater than 10's of years.

```

4284 This class does not require Generation to be unique across instances of other classes nor across
 4285 instances of the same class that have different keys. Generation shall be different across power
 4286 cycles, resets, or reboots if any of those actions results in an update. Generation may be
 4287 different across power cycles, resets, or reboots if those actions do not result in an update. If
 4288 the Generation property of an instance is non-null, and if any attempt to update the instance
 4289 includes the Generation property, then if it doesn't match the current value, the update shall
 4290 fail.

4291 The usage of this property is intended to be further specified by applicable management profiles.
 4292 Typically, a client will read the value of this property and then supply that value as input to
 4293 an operation that modifies the instance in some means. This may be via an explicit parameter in
 4294 an extrinsic method or via an embedded value in an extrinsic method or intrinsic operation.

4295 For example: a profile may require that an intrinsic instance modification supply the Generation
 4296 property and that it must match for the modification to succeed.</cim:Qualifier>

```

4297     <cim:Classorigin name="CIM_ManagedElement"/>
4298     </xs:appinfo>
4299     </xs:annotation>
4300 </xs:element>
4301 <xs:element name="SetPowerState_INPUT">
4302   <xs:complexType>
4303     <xs:sequence>
4304       <xs:element name="PowerState" nillable="true">
4305         <xs:complexType>
4306           <xs:simpleContent>
4307             <xs:restriction base="cim:cimUnsignedInt">
4308               <xs:enumeration value="1"/>
4309               <xs:enumeration value="2"/>
4310               <xs:enumeration value="3"/>
4311               <xs:enumeration value="4"/>
4312               <xs:enumeration value="5"/>
4313               <xs:enumeration value="6"/>
4314               <xs:enumeration value="7"/>
4315               <xs:enumeration value="8"/>
4316               <xs:anyAttribute namespace="##any" processContents="lax"/>
4317             </xs:restriction>
4318           </xs:simpleContent>
4319         </xs:complexType>
4320       </xs:element>
4321       <xs:element name="Time" nillable="true" type="cim:cimDateTime"/>
4322     </xs:sequence>
4323   </xs:complexType>
4324 </xs:element>
4325 <xs:element name="SetPowerState_OUTPUT">
4326   <xs:complexType>
4327     <xs:sequence>
4328       <xs:element name="ReturnValue" nillable="true" type="cim:cimUnsignedInt"/>
4329     </xs:sequence>
4330   </xs:complexType>
4331 </xs:element>
4332 <xs:element name="RequestStateChange_INPUT">
4333   <xs:complexType>
4334     <xs:sequence>
4335       <xs:element name="RequestedState" nillable="true">
4336         <xs:complexType>
4337           <xs:simpleContent>
4338             <xs:restriction base="cim:cimAnySimpleType">
4339           </xs:restriction>
4340         </xs:complexType>
4341       </xs:element>
4342     </xs:sequence>
4343   </xs:complexType>
4344 </xs:element>

```



```

4340         <xs:union>
4341             <xs:simpleType>
4342                 <xs:restriction base="xs:unsignedShort">
4343                     <xs:enumeration value="2"/>
4344                     <xs:enumeration value="3"/>
4345                     <xs:enumeration value="4"/>
4346                     <xs:enumeration value="6"/>
4347                     <xs:enumeration value="7"/>
4348                     <xs:enumeration value="8"/>
4349                     <xs:enumeration value="9"/>
4350                     <xs:enumeration value="10"/>
4351                     <xs:enumeration value="11"/>
4352                 </xs:restriction>
4353             </xs:simpleType>
4354             <xs:simpleType>
4355                 <xs:union>
4356                     <xs:simpleType>
4357                         <xs:restriction base="xs:unsignedShort">
4358                             <xs:maxInclusive value="1"/>
4359                         </xs:restriction>
4360                     </xs:simpleType>
4361                     <xs:simpleType>
4362                         <xs:restriction base="xs:unsignedShort">
4363                             <xs:enumeration value="5"/>
4364                         </xs:restriction>
4365                     </xs:simpleType>
4366                     <xs:simpleType>
4367                         <xs:restriction base="xs:unsignedShort">
4368                             <xs:minInclusive value="12"/>
4369                             <xs:maxInclusive value="32767"/>
4370                         </xs:restriction>
4371                     </xs:simpleType>
4372                 </xs:union>
4373             </xs:simpleType>
4374             <xs:simpleType>
4375                 <xs:restriction base="xs:unsignedShort">
4376                     <xs:minInclusive value="32768"/>
4377                     <xs:maxInclusive value="65535"/>
4378                 </xs:restriction>
4379             </xs:simpleType>
4380         </xs:union>
4381     </xs:simpleType>
4382     <xs:anyAttribute namespace="##any" processContents="lax"/>
4383 </xs:restriction>
4384 </xs:simpleContent>
4385 </xs:complexType>
4386 </xs:element>
4387 <xs:element name="TimeoutPeriod" nillable="true" type="cim:cimDateTime"/>
4388 </xs:sequence>
4389 </xs:complexType>
4390 </xs:element>
4391 <xs:element name="RequestStateChange_OUTPUT">
4392     <xs:complexType>

```

```

4393 <xs:sequence>
4394   <xs:element name="Job" nillable="true" type="cim:cimReference"/>
4395   <xs:element name="ReturnValue" nillable="true">
4396     <xs:complexType>
4397       <xs:simpleContent>
4398         <xs:restriction base="cim:cimAnySimpleType">
4399           <xs:simpleType>
4400             <xs:union>
4401               <xs:simpleType>
4402                 <xs:restriction base="xs:unsignedInt">
4403                   <xs:enumeration value="0"/>
4404                   <xs:enumeration value="1"/>
4405                   <xs:enumeration value="2"/>
4406                   <xs:enumeration value="3"/>
4407                   <xs:enumeration value="4"/>
4408                   <xs:enumeration value="5"/>
4409                   <xs:enumeration value="6"/>
4410                 </xs:restriction>
4411               </xs:simpleType>
4412               <xs:simpleType>
4413                 <xs:union>
4414                   <xs:simpleType>
4415                     <xs:restriction base="xs:unsignedInt">
4416                       <xs:minInclusive value="7"/>
4417                       <xs:maxInclusive value="4095"/>
4418                     </xs:restriction>
4419                   </xs:simpleType>
4420                   <xs:simpleType>
4421                     <xs:restriction base="xs:unsignedInt">
4422                       <xs:minInclusive value="32769"/>
4423                     </xs:restriction>
4424                   </xs:simpleType>
4425                 </xs:union>
4426               </xs:simpleType>
4427               <xs:simpleType>
4428                 <xs:restriction base="xs:unsignedInt">
4429                   <xs:enumeration value="4096"/>
4430                   <xs:enumeration value="4097"/>
4431                   <xs:enumeration value="4098"/>
4432                   <xs:enumeration value="4099"/>
4433                 </xs:restriction>
4434               </xs:simpleType>
4435               <xs:simpleType>
4436                 <xs:restriction base="xs:unsignedInt">
4437                   <xs:minInclusive value="4100"/>
4438                   <xs:maxInclusive value="32767"/>
4439                 </xs:restriction>
4440               </xs:simpleType>
4441               <xs:simpleType>
4442                 <xs:restriction base="xs:unsignedInt">
4443                   <xs:minInclusive value="32768"/>
4444                   <xs:maxInclusive value="65535"/>
4445                 </xs:restriction>

```

```

4446         </xs:simpleType>
4447         </xs:union>
4448     </xs:simpleType>
4449     <xs:anyAttribute namespace="##any" processContents="lax" />
4450 </xs:restriction>
4451 </xs:simpleContent>
4452 </xs:complexType>
4453 </xs:element>
4454 </xs:sequence>
4455 </xs:complexType>
4456 </xs:element>
4457 <xs:element name="CIM_ComputerSystem" type="class:CIM_ComputerSystem_Type">
4458     <xs:annotation>
4459         <xs:appinfo source="http://schemas.dmtf.org/wbem/wscim/1/common">
4460             <cim:Class name="CIM_ComputerSystem" superclass="CIM_System" />
4461             <cim:Qualifier name="Version" type="cim:cimString">2.24.0</cim:Qualifier>
4462             <cim:Qualifier name="UMLPackagePath"
4463 type="cim:cimString">CIM::System::SystemElements</cim:Qualifier>
4464             <cim:Qualifier name="Description" type="cim:cimString">A class derived from System that
4465 is a special collection of ManagedSystemElements. This collection is related to the providing of
4466 compute capabilities and MAY serve as an aggregation point to associate one or more of the
4467 following elements: FileSystem, OperatingSystem, Processor and Memory (Volatile and/or
4468 NonVolatile Storage).</cim:Qualifier>
4469         </xs:appinfo>
4470     </xs:annotation>
4471 </xs:element>
4472 <xs:complexType name="CIM_ComputerSystem_Type">
4473     <xs:sequence>
4474         <xs:element maxOccurs="unbounded" minOccurs="0" ref="class:AvailableRequestedStates" />
4475         <xs:element minOccurs="0" ref="class:Caption" />
4476         <xs:element minOccurs="0" ref="class:CommunicationStatus" />
4477         <xs:element ref="class:CreationClassName" />
4478         <xs:element maxOccurs="unbounded" minOccurs="0" ref="class:Dedicated" />
4479         <xs:element minOccurs="0" ref="class:Description" />
4480         <xs:element minOccurs="0" ref="class:DetailedStatus" />
4481         <xs:element minOccurs="0" ref="class:ElementName" />
4482         <xs:element minOccurs="0" ref="class:EnabledDefault" />
4483         <xs:element minOccurs="0" ref="class:EnabledState" />
4484         <xs:element minOccurs="0" ref="class:Generation" />
4485         <xs:element minOccurs="0" ref="class:HealthState" />
4486         <xs:element maxOccurs="unbounded" minOccurs="0" ref="class:IdentifyingDescriptions" />
4487         <xs:element minOccurs="0" ref="class:InstallDate" />
4488         <xs:element minOccurs="0" ref="class:InstanceID" />
4489         <xs:element ref="class:Name" />
4490         <xs:element minOccurs="0" ref="class:NameFormat" />
4491         <xs:element minOccurs="0" ref="class:OperatingStatus" />
4492         <xs:element maxOccurs="unbounded" minOccurs="0" ref="class:OperationalStatus" />
4493         <xs:element maxOccurs="unbounded" minOccurs="0" ref="class:OtherDedicatedDescriptions" />
4494         <xs:element minOccurs="0" ref="class:OtherEnabledState" />
4495         <xs:element maxOccurs="unbounded" minOccurs="0" ref="class:OtherIdentifyingInfo" />
4496         <xs:element maxOccurs="unbounded" minOccurs="0" ref="class:PowerManagementCapabilities" />
4497         <xs:element minOccurs="0" ref="class:PrimaryOwnerContact" />
4498         <xs:element minOccurs="0" ref="class:PrimaryOwnerName" />
4499         <xs:element minOccurs="0" ref="class:PrimaryStatus" />
4500         <xs:element minOccurs="0" ref="class:RequestedState" />

```

```
4501 <xs:element minOccurs="0" ref="class:ResetCapability"/>
4502 <xs:element maxOccurs="unbounded" minOccurs="0" ref="class:Roles"/>
4503 <xs:element minOccurs="0" ref="class:Status"/>
4504 <xs:element maxOccurs="unbounded" minOccurs="0" ref="class:StatusDescriptions"/>
4505 <xs:element minOccurs="0" ref="class:TimeOfLastStateChange"/>
4506 <xs:element minOccurs="0" ref="class:TransitioningToState"/>
4507 <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##other" processContents="lax"/>
4508 </xs:sequence>
4509 <xs:anyAttribute namespace="##any" processContents="lax"/>
4510 </xs:complexType>
4511 </xs:schema>
```

4512

4513

**ANNEX E
(Informative)****Change Log**

Version	Date	Description
1.0.0	2008-05-22	Released as Final Standard
1.0.1	2009-04-21	Released as DMTF Standard, with the following changes: <ul style="list-style-type: none">• Clarified ambiguous references to WS-Addressing specifications in clause 3• Clarified rules for collation sequence of properties in subclause 9.3.1 and in the new ANNEX C
1.0.2	2010-02-08	Released as DMTF Standard, with the following changes: <ul style="list-style-type: none">• Changed the representation of octet strings in subclause 9.2.4 to remove the length representation
1.1.0	2011-06-30	DMTF Standard

4514
4515
4516
4517
4518

4519

4520

Bibliography

4521 DMTF DSP4009, *Process for publishing XML schema, XML documents and XSLT stylesheets 1.0*,
4522 http://www.dmtf.org/standards/published_documents/DSP4009_1.0.pdf

4523