



1
2
3
4

Document Number: DSP1002

Date: 2013-06-13

Version: 2.1.0a

5 **Diagnostics Profile**

Information for Work-in-Progress version:

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

It expires on: 2013-09-27

Provide any comments through the DMTF Feedback Portal:

<http://www.dmtf.org/standards/feedback>

6 **Document Type: Specification**
7 **Document Status: Work in Progress**
8 **Document Language: en-US**

9 Copyright Notice

10 Copyright © 2006, 2009, 2010, 2013 Distributed Management Task Force, Inc. (DMTF). All rights
11 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26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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	7
35	Introduction.....	8
36	1 Scope	9
37	2 Normative references	9
38	3 Terms and definitions	10
39	4 Symbols and abbreviated terms.....	10
40	5 Synopsis	11
41	6 Description	12
42	7 Implementation.....	14
43	7.1 CIM_DiagnosticTest.....	14
44	7.2 CIM_AvailableDiagnosticService	16
45	7.3 CIM_DiagnosticServiceCapabilities	17
46	7.4 CIM_DiagnosticSettingData.....	23
47	7.5 CIM_DiagnosticLog.....	26
48	7.6 CIM_DiagnosticRecord.....	26
49	7.7 CIM_ServiceComponent.....	27
50	7.8 Diagnostics Profile Indications support.....	27
51	7.9 Diagnostics alert indications and standard messages.....	29
52	8 Methods.....	46
53	8.1 CIM_DiagnosticService.RunDiagnosticService() extrinsic method	46
54	8.2 CIM_Log.ClearLog() extrinsic method	47
55	8.3 CIM_HelpService.GetHelp() extrinsic method	48
56	8.4 Profile conventions for operations	48
57	8.5 CIM_DiagnosticTest.....	49
58	8.6 CIM_AvailableDiagnosticService	49
59	8.7 CIM_ServiceAffectsElement	50
60	8.8 CIM_SoftwareIdentity.....	50
61	8.9 CIM_ElementSoftwareIdentity	50
62	8.10 CIM_HelpService	51
63	8.11 CIM_ServiceAvailableToElement	51
64	8.12 CIM_DiagnosticSettingData.....	51
65	8.13 CIM_DiagnosticServiceCapabilities	52
66	8.14 CIM_ElementCapabilities	52
67	8.15 CIM_ElementSettingData	53
68	8.16 CIM_DiagnosticLog.....	53
69	8.17 CIM_UseOfLog	53
70	8.18 CIM_DiagnosticServiceRecord.....	54
71	8.19 CIM_DiagnosticCompletionRecord.....	54
72	8.20 CIM_DiagnosticSettingDataRecord	55
73	8.21 CIM_LogManagesRecord.....	56
74	8.22 CIM_RecordAppliesToElement	56
75	8.23 CIM_CorrespondingSettingDataRecord	56
76	8.24 CIM_ServiceComponent.....	56
77	9 Use cases.....	58
78	9.1 Profile conformance	58
79	9.2 Use case summary	59
80	9.3 Diagnostic services object diagram	61
81	9.4 Discover available diagnostics.....	62
82	9.5 Configure diagnostic	63
83	9.6 Execute and control diagnostic.....	65
84	9.7 Discover diagnostic executions	68
85	9.8 Discover diagnostic results (In Progress and Final)	70

86	10	CIM Elements	76
87	10.1	CIM_AvailableDiagnosticService	79
88	10.2	CIM_CorrespondingSettingDataRecord (DiagnosticServiceRecord)	80
89	10.3	CIM_CorrespondingSettingDataRecord (DiagnosticCompletionRecord)	80
90	10.4	CIM_DiagnosticCompletionRecord	81
91	10.5	CIM_DiagnosticLog	82
92	10.6	CIM_DiagnosticServiceCapabilities	82
93	10.7	CIM_DiagnosticServiceRecord	83
94	10.8	CIM_DiagnosticSettingData (Default)	85
95	10.9	CIM_DiagnosticSettingData (Client)	87
96	10.10	CIM_DiagnosticSettingDataRecord	89
97	10.11	CIM_DiagnosticTest	90
98	10.12	CIM_ElementCapabilities	90
99	10.13	CIM_ElementSettingData (JobSettingData)	90
100	10.14	CIM_ElementSettingData (DiagnosticSettingData)	91
101	10.15	CIM_ElementSoftwareIdentity	91
102	10.16	CIM_FilterCollection	92
103	10.17	CIM_HelpService	92
104	10.18	CIM_HostedService	93
105	10.19	CIM_IndicationFilter	93
106	10.20	CIM_LogManagesRecord	94
107	10.21	CIM_MemberOfCollection	94
108	10.22	CIM_OwningCollectionElement	94
109	10.23	CIM_RecordAppliesToElement	95
110	10.24	CIM_RegisteredProfile	95
111	10.25	CIM_ServiceAffectsElement	95
112	10.26	CIM_ServiceAvailableToElement	96
113	10.27	CIM_ServiceComponent	96
114	10.28	CIM_SoftwareIdentity	97
115	10.29	CIM_UseOfLog	97
116		ANNEX A (informative) Change log	98
117		Bibliography	99
118			

119 **Figures**

120 Figure 1 – Diagnostics Profile: Class diagram 13
 121 Figure 2 – Registered profile 59
 122 Figure 3 – Diagnostic services object diagram 61
 123 Figure 4 – Job example 66
 124 Figure 5 – Diagnostic logging object diagram 71
 125

126 **Tables**

127 Table 1 – Related profiles 12
 128 Table 2 – RunDiagnosticService() method: Return code values 47
 129 Table 3 – RunDiagnosticService() method: Parameters 47
 130 Table 4 – ClearLog() method: Return code values 47
 131 Table 5 – GetHelp() method: Return code values 48
 132 Table 6 – GetHelp() method: Parameters 48
 133 Table 7 – Operations: CIM_DiagnosticTest 49
 134 Table 8 – Operations: CIM_AvailableDiagnosticService 49
 135 Table 9 – Operations: CIM_ServiceAffectsElement 50
 136 Table 10 – Operations: CIM_SoftwareIdentity 50
 137 Table 11 – Operations: CIM_ElementSoftwareIdentity 50
 138 Table 12 – Operations: CIM_HelpService 51
 139 Table 13 – Operations: CIM_ServiceAvailableToElement 51
 140 Table 14 – Operations: CIM_DiagnosticSettingData 52
 141 Table 15 – Operations: CIM_DiagnosticServiceCapabilities 52
 142 Table 16 – Operations: CIM_ElementCapabilities 52
 143 Table 17 – Operations: CIM_ElementSettingData 53
 144 Table 18 – Operations: CIM_DiagnosticLog 53
 145 Table 19 – Operations: CIM_UseOfLog 54
 146 Table 20 – Operations: CIM_DiagnosticServiceRecord 54
 147 Table 21 – Operations: CIM_DiagnosticCompletionRecord 55
 148 Table 22 – Operations: CIM_DiagnosticSettingDataRecord 55
 149 Table 23 – Operations: CIM_LogManagesRecord 56
 150 Table 24 – Operations: CIM_RecordAppliesToElement 56
 151 Table 25 – Operations: CIM_CorrespondingSettingDataRecord 56
 152 Table 26 – Operations: CIM_ServiceComponent 57
 153 Table 27 – Diagnostics Profile use cases 59
 154 Table 28 – CIM Elements: Diagnostics Profile 76
 155 Table 29 – Class: CIM_AvailableDiagnosticService 79
 156 Table 30 – Class: CIM_CorrespondingSettingDataRecord 80
 157 Table 31 – Class: CIM_CorrespondingSettingDataRecord 80
 158 Table 32 – Class: CIM_DiagnosticCompletionRecord 81
 159 Table 33 – Class: CIM_DiagnosticLog 82
 160 Table 34 – Class: CIM_DiagnosticServiceCapabilities 82
 161 Table 35 – Class: CIM_DiagnosticServiceRecord 83
 162 Table 36 – Class: CIM_DiagnosticSettingData 85
 163 Table 37 – Class: CIM_DiagnosticSettingData 87
 164 Table 38 – Class: CIM_DiagnosticSettingDataRecord 89

165	Table 39 – Class: CIM_DiagnosticTest.....	90
166	Table 40 – Class: CIM_ElementCapabilities.....	90
167	Table 41 – Class: CIM_ElementSettingData	91
168	Table 42 – Class: CIM_ElementSettingData	91
169	Table 43 – Class: CIM_ElementSoftwareIdentity	91
170	Table 44 - Class: CIM_FilterCollection	92
171	Table 45 – Class: CIM_HelpService	92
172	Table 46 – Class: CIM_HostedService	93
173	Table 47 - Class: CIM_IndicationFilter	93
174	Table 48 – Class: CIM_LogManagesRecord.....	94
175	Table 49 - Class: CIM_MemberOfCollection	94
176	Table 50 - Class: CIM_OwningCollectionElement.....	94
177	Table 51 – Class: CIM_RecordAppliesToElement	95
178	Table 52 – Class: CIM_RegisteredProfile	95
179	Table 53 – Class: CIM_ServiceAffectsElement	95
180	Table 54 – Class: CIM_ServiceAvailableToElement	96
181	Table 55 – Class: CIM_ServiceComponent.....	96
182	Table 56 – Class: CIM_SoftwareIdentity.....	97
183	Table 57 – Class: CIM_UseOfLog	97
184		

185

Foreword

186 The *Diagnostics Profile* (DSP1002) was prepared by the DMTF.

187 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
188 management and interoperability. For information about the DMTF, see <http://www.dmtf.org>.

189 **Acknowledgments**

190 The DMTF acknowledges the following individuals for their contributions to this document:

- 191 • Rodney Brown – IBM Corporation
- 192 • Carl Chan – WBEM Solutions, Inc.
- 193 • Peter Lamanna – EMC Corporation
- 194 • Mike Walker – Storage Networking Industry Association

195

196

197

Introduction

198 A *profile* is a collection of Common Information Model (CIM) elements and behavior rules that represents
199 a specific area of management. The purpose of a profile is to ensure interoperability in the use of Web-
200 Based Enterprise Management (WBEM) services for a specific subset of the Distributed Management
201 Task Force (DMTF) CIM schema for a specific management area — in this case, diagnostics.

202 Diagnostics is a critical component of systems management. Diagnostic services are used in problem
203 containment to maintain availability, achieve fault isolation for system recovery, establish system integrity
204 during boot, increase system reliability, and perform routine proactive system verification. The goal of the
205 Common Diagnostic Model (CDM) is to define industry-standard building blocks, based on and consistent
206 with the DMTF CIM, that enables seamless integration of vendor-supplied diagnostic services into
207 systems management frameworks, for example SAN management frameworks.

208 The CDM is an architecture and methodology for exposing system diagnostic instrumentation through the
209 CIM standard interfaces.

210 The ability to transparently run diagnostic tests and exercisers while the user operating system is
211 functional (no reboot required) may significantly contribute to the reduction of Total Cost of Ownership
212 (TCO) and will also lower warranty costs by reducing the return of defect-free parts for service. This
213 functionality is referred to as *OS-Present Diagnostics* (also known as On-line Diagnostics and Concurrent
214 Diagnostics).

215 A primary objective of the CDM is to standardize the interfaces that diagnostic developers create for their
216 OS-Present Diagnostics in the operating environment, making the diagnostics accessible to all
217 applications that query CIM for diagnostic data or register with CIM to execute diagnostic methods and
218 receive results.

219 Standardization of these interfaces means that clients, implementations, and tests gain a certain degree
220 of portability and, in many cases, need only be written once to satisfy multiple environments and
221 platforms. OEMs can differentiate their diagnostic offerings by how effectively their applications use the
222 information and capabilities available through CIM to maintain and service their systems.

223 Reduced cost through standardization is accompanied by the initial investment of coding to a new
224 interface.

225

Diagnostics Profile

1 Scope

227 The information in this specification should be sufficient for a provider or consumer of this data to identify
228 unambiguously the classes, properties, methods, and values that shall be instantiated and manipulated to
229 represent and manage the diagnostic service components of systems and subsystems that are modeled
230 using the DMTF CIM core and extended model definitions.

231 The target audience for this specification is implementers who are developing implementations or
232 consumers of management interfaces that represent the functionality described in this document.

2 Normative references

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

238 DMTF DSP0004, *CIM Infrastructure Specification 2.6*,
239 http://www.dmtf.org/standards/published_documents/DSP0004_2.6.pdf

240 DMTF DSP0200, *CIM Operations over HTTP 1.3*,
241 http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf

242 DMTF DSP0215, *SM Managed Element Addressing Specification (SM ME Addressing) 1.0*,
243 http://www.dmtf.org/sites/default/files/standards/documents/DSP0215_1.0.pdf

244 DMTF DSP1001, *Management Profile Specification Usage Guide 1.0*,
245 http://www.dmtf.org/standards/published_documents/DSP1001_1.0.pdf

246 DMTF DSP1004, *Base Server Profile 1.0*,
247 http://www.dmtf.org/standards/published_documents/DSP1004_1.0.pdf

248 DMTF DSP1033, *Profile Registration Profile 1.0*,
249 http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf

250 DMTF DSP1054, *Indications Profile 1.2*,
251 http://dmtf.org/sites/default/files/standards/documents/DSP1054_1.2.pdf

252 DMTF DSP1119, *Diagnostic Job Control Profile 1.0.0b*,
253 http://dmtf.org/sites/default/files/standards/documents/DSP1119_1.0.pdf

254 DMTF DSP8055, *Diagnostics Message Registry 1.0.0a*,
255 http://dmtf.org/sites/default/files/standards/documents/DSP8055_1.0a.xml

256 IETF RFC5234, *ABNF: Augmented BNF for Syntax Specifications, January 2008*,
257 <http://tools.ietf.org/html/rfc5234>

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

260

261 **3 Terms and definitions**

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

264 The terms "shall" ("required"), "shall not," "should" ("recommended"), "should not" ("not recommended"),
265 "may," "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described
266 in [ISO/IEC Directives, Part 2](#), Annex H. The terms in parenthesis are alternatives for the preceding term,
267 for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that
268 [ISO/IEC Directives, Part 2](#), Annex H specifies additional alternatives. Occurrences of such additional
269 alternatives shall be interpreted in their normal English meaning.

270 The terms "clause," "subclause," "paragraph," and "annex" in this document are to be interpreted as
271 described in [ISO/IEC Directives, Part 2](#), Clause 5.

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

275 The terms defined in [DSP0004](#), [DSP0200](#), [DSP1001](#), and [DSP1033](#) apply to this document. For the
276 purposes of this document, the following terms and definitions also apply.

277 **3.1**

278 **conditional**

279 indicates requirements to be followed strictly in order to conform to the document when the specified CIM
280 testable conditions are met

281 **3.2**

282 **mandatory**

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

285 **3.3**

286 **optional**

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

288 **4 Symbols and abbreviated terms**

289 The following abbreviations are used in this document.

290 **4.1**

291 **CDM**

292 Common Diagnostic Model

293 **4.2**

294 **CIM**

295 Common Information Model

296 **4.3**

297 **CIMOM**

298 CIM Object Manager

- 299 **4.4**
- 300 **CRU**
- 301 Customer Replaceable Unit
- 302 **4.5**
- 303 **FRU**
- 304 Field Replaceable Unit
- 305 **4.6**
- 306 **ME**
- 307 Managed Element
- 308 **4.7**
- 309 **MOF**
- 310 Managed Object Format
- 311 **4.8**
- 312 **PD**
- 313 Problem Determination
- 314 **4.9**
- 315 **PFA**
- 316 Predictive Failure Analysis
- 317 **4.10**
- 318 **SAN**
- 319 Storage Area Network
- 320 **4.11**
- 321 **WBEM**
- 322 Web-Based Enterprise Management

323 **5 Synopsis**

324 **Profile Name:** Diagnostics Profile

325 **Version:** 2.1.0

326 **Organization:** DMTF

327 **CIM schema version:** 2.36

328 **Central Class:** CIM_DiagnosticTest

329 **Scoping Class:** CIM_ComputerSystem

330 The *Diagnostics Profile* extends the management capability of referencing profiles by adding the
 331 capability to run diagnostic services in a managed system. This profile includes a specification of the
 332 Diagnostic Test Service, its configuration, its associated capabilities, its logging mechanisms, and its
 333 profile registration information.

334 Table 1 identifies profiles on which this profile has a dependency.

335 CIM_DiagnosticTest shall be the Central Class of this profile. The instance of CIM_DiagnosticTest shall
 336 be the Central Instance of this profile. CIM_ComputerSystem shall be the Scoping Class of this profile.

337 The instance of CIM_ComputerSystem with which the Central Instance is associated through an instance
 338 of CIM_HostedService shall be the Scoping Instance of this profile.

339 **Table 1 – Related profiles**

Profile Name	Organization	Version	Relationship	Behavior
Profile Registration	DMTF	1.0	Mandatory	
Diagnostic Job Control	DMTF	1.0.0b	Mandatory	
Indications	DMTF	1.2.2	Mandatory	

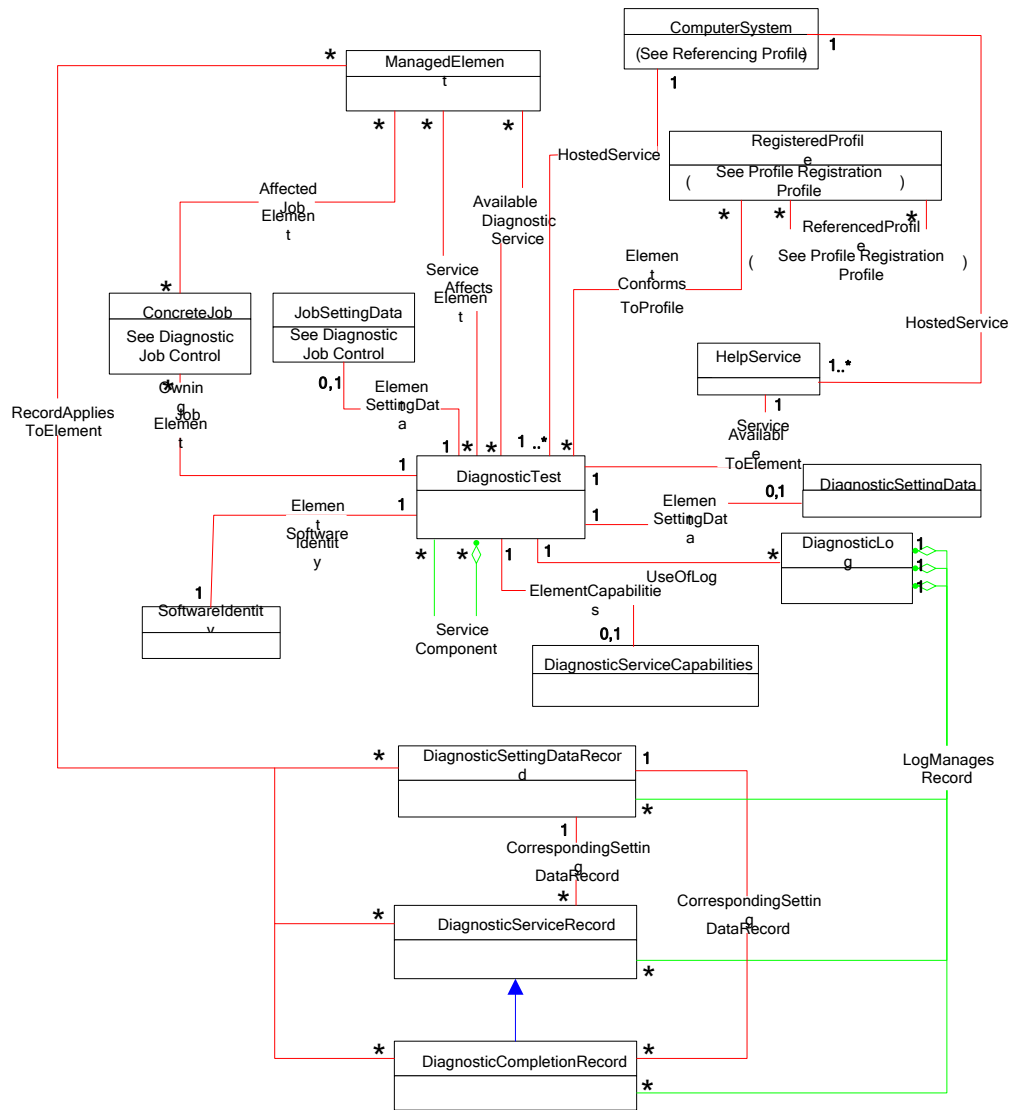
340

341 **6 Description**

342 This profile describes the CIM schema extensions that compose the Common Diagnostic Model (CDM)
 343 and provides guidelines for the development of diagnostic clients and implementations that will promote
 344 seamless integration of option diagnostics into Problem Determination and Systems Management
 345 applications. Using this profile as a guide, WBEM clients can discover diagnostic services that have been
 346 installed on the system and invoke these services to run on their respective devices. The client can
 347 monitor the progress of the service, obtain and modify the status of the service, and query for results.

348 The architecture of the CDM is described in the [CIM Diagnostic Model White Paper](#). This profile is a
 349 normative presentation of the model described in the white paper, and it suggests implementation
 350 techniques that will result in the highest degree of interoperability. It is targeted at developers of
 351 diagnostic applications (WBEM clients) and hardware instrumentation (for the WBEM server) to help them
 352 understand the spirit and intent of the CDM.

353 Figure 1 presents the class schema for the *Diagnostics Profile*. For simplicity, the prefix CIM_ has been
 354 removed from the names of the classes.



355

356

357

Figure 1 – Diagnostics Profile: Class diagram

358 **7 Implementation**

359 This clause details the requirements related to the arrangement of instances and their properties for
360 implementations of this profile.

361 The *Diagnostics Profile* consists of definitions for classes related to the CIM_DiagnosticService class,
362 such as CIM_DiagnosticTest, CIM_DiagnosticSettingData, and CIM_DiagnosticServiceCapabilities. It
363 also defines the CIM_DiagnosticLog class and its related classes, CIM_DiagnosticRecord,
364 CIM_DiagnosticServiceRecord, and CIM_DiagnosticSettingDataRecord. Requirements for propagating
365 and formulating certain properties of these classes and their parents are discussed in this clause.
366 Required methods are listed in clause 8, and properties are listed in clause 10.

367 **7.1 CIM_DiagnosticTest**

368 CIM_DiagnosticTest is the only defined subclass of CIM_DiagnosticService. CIM_DiagnosticTest inherits
369 the RunDiagnosticService() method, which is used to execute a diagnostic test on a managed element.

370 Each diagnostic test shall be represented by an instance of either the CIM_DiagnosticTest or a subclass.
371 Note that a test that actually packages multiple subtests shall also be represented by such an instance
372 and shall set the IsPackage characteristic for that instance (see 7.1.3.5).

373 An implementation may use

- 374 • an instance of CIM_DiagnosticTest for each test
- 375 • an instance of a single subclass (for example, ST_DiskDiagnosticTest) for each test
- 376 • a different subclass and its instance (for example, ST_DiskDiagnosticSelfTest,
377 ST_DiskDiagnosticRWVTest) for each test

378 The same implementation may use a combination of the preceding approaches.

379 **7.1.1 CIM_DiagnosticTest.Name**

380 The Name property uniquely identifies the service and provides an indication of the functionality that is
381 managed. The value of the Name property shall be unique and should indicate the nature of the service
382 (for example, EjectTest).

383 **7.1.2 CIM_DiagnosticTest.ElementName**

384 The ElementName property shall be used to provide a user-friendly name for the service. This name shall
385 be used by clients to identify the service to the user.

386 **7.1.3 CIM_DiagnosticTest.Characteristics**

387 This clause defines the values of the Characteristics property.

388 **7.1.3.1 Is Exclusive (value=2)**

389 Use this value to indicate that only one instance of the diagnostic test may be running at one time, even if
390 more than one target device exists.

391 If the test can run on multiple target devices, but only one instance per device, use
392 CIM_AvailableDiagnosticService.IsExclusiveForMSE.

393 7.1.3.2 Is Interactive (value=3)

394 Use this value to indicate that the test requires some interaction with the client at the system under test
395 (for example, when media is required in a device for the test to run).

396 For a description of how a client application interacts with a diagnostic test, see the *Diagnostics Job*
397 *Control Profile* ([DSP1119](#)).

398 7.1.3.3 Is Destructive (value=4)

399 Use this value to indicate that the test has the potential for destroying data, permanently altering the
400 state, or reconfiguring the device.

401 7.1.3.4 Is Risky (value=5)

402 Use this value to indicate that data loss, state change, or reconfiguration may occur if the test is
403 interrupted. For example, a test saves some device data or configuration, changes the device state,
404 performs some operation, and then restores the saved data. If this process is interrupted, the device may
405 be left in an altered state.

406 7.1.3.5 Is Package (value=6)

407 Use this value to indicate that the test is actually a set of lower-level diagnostics that are packaged
408 together by the test. This packaging is implemented by the test, not aggregated by CIM. Information and
409 results associated with the individual tests in the package may be requested by using the Subtests value
410 in the CIM_DiagnosticSettingData.LogOptions array.

411 If the lower-level diagnostics are themselves CIM_DiagnosticTest instances, the packaging test shall be
412 associated to those lower-level diagnostics through an instance of the CIM_ServiceComponent
413 association. See 7.8.

414 7.1.3.6 Reserved (value=7)

415 This value originally contained "Supports PercentOfTestCoverage", which was deprecated and added to
416 the CIM_DiagnosticServiceCapabilities class.

417 7.1.3.7 Is Synchronous (value=8)

418 Use this value to indicate that this diagnostic service will be completed before the
419 RunDiagnosticService() method returns to the caller. A job is still created that the client may access for
420 accounting purposes, but the ability to track the progress and status of the job are lost. Additionally, in
421 certain environments, the client may be "blocked" from further action until the service is completed.
422 Development of synchronous diagnostic services is not recommended.

423 7.1.3.8 Media Required (value=9)

424 Use this value to indicate that media must be inserted into the device to perform the service.

425 7.1.3.9 Additional Hardware Required (value=10)

426 Use this value to indicate that some additional hardware (for example, a wrap plug) must be installed to
427 perform the service.

428 7.1.4 CIM_DiagnosticTest.TestTypes

429 The TestTypes is an optional array property that provides a high-level description of the nature of the test
430 If supplied, the possible values are 1 (Other), 2 (Functional), 3 (Stress), 4 (Health Check), 5 (Access
431 Test), or 6 (Media Verify).

432 **7.1.5 OtherTestTypesDescriptions**

433 The “Other” TestType is provided for vendor-specific service modes. If this property is specified, the
434 OtherTestTypesDescriptions shall have at least one value.

435 **7.1.6 Looping tests**

436 Looping tests or groups of tests are useful for detecting intermittent faults. The client, implementation, or
437 test may control looping, and the method chosen depends on many factors, a few of which follow:

- 438 • A client may want to loop a test that does not support looping.
- 439 • An implementation may choose to support looping even though its tests do not.
- 440 • A stress test may, by its nature, want to repeat a certain operation multiple times.

441 Looping in the implementation and test is under control of the LoopControl() and LoopControlParameter()
442 properties of the CIM_DiagnosticSettingData class. These properties are used to specify the number of
443 iterations in the loop, either directly or through a termination condition. If more than one control is set, the
444 first one that reaches its condition terminates the loop.

445 Looping in the client is entirely under the control of the client and would generally not affect the
446 CIM_DiagnosticSettingData object.

447 **NOTE** A remote client may incur network delays and CIMOM delays during each iteration of its loop, and this is not
448 an effective way to stress a device.

449 It is recommended that all diagnostic tests support looping. Exceptions exist where looping a test leads to
450 an undesirable condition (for example, a risky test, certain user interactions, or excessive mechanical
451 wear).

452 **7.1.7 Test effectiveness**

453 Although the focus of this profile is use of the CIM schema, the CDM includes the notion of test
454 effectiveness. A perfectly implemented CDM implementation coupled with an ineffective test is not very
455 useful.

456 Diagnostic tests should provide support for all properties in the CIM_DiagnosticSettingData class.

457 The QuickMode property of the CIM_DiagnosticSettings class shall be supported for “long-running” tests
458 (that is, tests with running times in excess of what would be considered compatible with a quick system
459 “health check” of a few minutes). QuickMode need not be supported for interactive, risky, or destructive
460 tests because these tests would not be useful as a health check.

461 **NOTE** QuickMode is distinct from PercentOfTestCoverage in that it is a Boolean property that may be set by a
462 client without any particular knowledge of the test. Use of PercentOfTestCoverage requires that the client be aware of
463 the effects and expected outcome of this “throttling” setting control.

464 **7.2 CIM_AvailableDiagnosticService**

465 An instance of CIM_AvailableDiagnosticService shall associate a managed element with a diagnostic
466 service that is available for that element. This instance is the means by which clients discover the
467 diagnostic services that are installed for a particular managed element.

468 **7.2.1 CIM_AvailableDiagnosticService.EstimatedDurationOfService**

469 All tests shall attempt to accurately set the EstimatedDurationOfService property. As stated in the MOF
470 file for this class, this property is an estimation of magnitude, not absolute time, and is to be used as a
471 guide for the client.

472 The CIM_DiagnosticSettingData.LoopControl property allows a client to indicate how long a test should
473 run. Tests should use their default values for the LoopControl properties when determining a value for
474 EstimatedDurationOfService.

475 Interactive tests have an additional complication because their test execution depends on the responses
476 from the user. However, this type of test is not much different than a test whose execution depends on
477 information from a device and the response time of the hardware, or even on how much CPU time or
478 other system resources are allocated to the test. Interactive tests should assume a user response time. If
479 a test cannot reasonably determine an EstimatedDurationOfService value (for example, a completely
480 interactive test that does not know anything about what it will do until a user tells it what tests to run), it
481 can set the value to 0 (Unknown).

482 **7.2.2 CIM_AvailableDiagnosticService.EstimatedDurationQualifier**

483 The EstimatedDurationQualifier property allows for more accurate quantification of the value specified for
484 the EstimatedDurationOfService property. For example, if EstimatedDurationOfService has the value 2
485 (Seconds) and EstimatedDurationQualifier has a value of 20, the service has an estimated duration of 20
486 seconds. This property should be implemented if further quantification is possible. In contrast, if
487 EstimatedDurationOfService has the value 0 (Unknown), EstimatedDurationQualifier may be NULL.

488 **7.3 CIM_DiagnosticServiceCapabilities**

489 A diagnostic service publishes its support for various options by using
490 CIM_DiagnosticServiceCapabilities. A client uses CIM_ElementCapabilities to find the diagnostic service
491 capabilities. CIM_DiagnosticServiceCapabilities and CIM_DiagnosticSettingData are closely related and
492 have similar properties. The settings used to control the execution of a diagnostic test cannot specify
493 unsupported capabilities.

494 **7.3.1 CIM_DiagnosticServiceCapabilities.SupportedServiceModes**

495 This property identifies the service modes supported by the DiagnosticTest. Multiple entries may be
496 provided in the SupportedServiceModes. If service modes are supported, they shall be published by
497 using this property. That is, a test may support none, one, or many of the service modes. The service
498 modes that may be specified are 1 (Other), 2 (PercentOfTestCoverage), 3 (QuickMode), 4 (HaltOnError),
499 5 (ResultPersistence), 6 (NonDestructive), 7 (No Service Modes).

500 **7.3.1.1 Other**

501 The “Other” service mode is provided for vendor-specific service modes. If this mode is specified, the
502 OtherServiceModesDescriptions shall have at least one value.

503 NOTE If “Other” is specified, the implementing vendor should also extend the CIM_DiagnosticSettingData class to
504 include any specification of support needed.

505 **7.3.1.2 PercentOfTestCoverage**

506 If this service mode is supported, the client may request the test to reduce its coverage to the specified
507 percentage set in the PercentOfTestCoverage property of the DiagnosticSettings parameter (an
508 embedded instance of CIM_DiagnosticSettingData) of the RunDiagnosticService() method. The effect of
509 the percentage specified is determined by the provider implementation.

510 **7.3.1.3 QuickMode**

511 If this service mode is supported, the client may request the test attempt to run in an accelerated manner
512 either by reducing the coverage or reducing the number of tests performed (as determined by the
513 provider implementation). The client requests this mode by specifying QuickMode=TRUE in the
514 QuickMode property of the DiagnosticSettings parameter (an embedded instance of
515 CIM_DiagnosticSettingData) of the RunDiagnosticService() method.

516 7.3.1.4 HaltOnError

517 If this service mode is supported, the client may request the test to halt after finding the first error. The
518 client requests this mode by specifying HaltOnError=TRUE in the HaltOnError property of the
519 DiagnosticSettings parameter (an embedded instance of CIM_DiagnosticSettingData) of the
520 RunDiagnosticService() method.

521 Depending on the test, it may make sense to have this mode set to FALSE to allow all errors to be
522 captured.

523 7.3.1.5 ResultPersistence

524 If this service mode is supported, the client may request how many seconds the records should persist
525 after test execution finishes. The client requests this mode by specifying the number of seconds in the
526 ResultPersistence property of the DiagnosticSettings parameter (an embedded instance of
527 CIM_DiagnosticSettingData) of the RunDiagnosticService() method. Supplying 0 (zero) indicates “no
528 persistence” and supplying 0xFFFFFFFF indicates “persist forever”.

529 If an implementation claims support for ResultPersistence, it shall support any value supplied in the
530 DiagnosticSettingData.ResultPersistence, except the 0xFFFFFFFF (“persist forever”) value. The
531 0xFFFFFFFF may or may not be supported. If an implementation cannot support all other values for
532 ResultPersistence, it shall not include ResultPersistence in its list of SupportedServiceModes.

533 7.3.1.6 NonDestructive

534 If this service mode is supported, the client may request the test only run nondestructive tests (as
535 determined by the provider implementation). The client requests this mode by specifying
536 NonDesctructive=TRUE in the NonDesctructive property of the DiagnosticSettings parameter (an
537 embedded instance of CIM_DiagnosticSettingData) of the RunDiagnosticService() method.

538 What constitutes a destructive test may vary depending on the device and how it is configured. For
539 example, if you are performing a random write test on a new disk drive that you have not put into service,
540 the test would not be destructive to any data or configuration. However, a random write test would be
541 destructive to a disk drive that is configured and in use in an array.

542 When a client specifies this mode, the client does not have to identify what subtests should not be run.
543 The provider will determine which subtests are destructive and not execute them.

544 7.3.2 CIM_DiagnosticServiceCapabilities.OtherSupportedServiceModesDescriptions

545 The OtherSupportedServiceModesDescriptions provide additional information for
546 SupportedServiceModes when the corresponding value is set to 1 ("Other"). This is intended for
547 vendor-specific extensions to the profile.

548 7.3.3 CIM_DiagnosticServiceCapabilities.SupportedLoopControl

549 This property identifies the loop controls supported by the DiagnosticTest. If looping is supported (see
550 7.1.6), its controls shall be published by using this property. Multiple entries may be provided in the
551 SupportedLoopControl. That is, a test may support none, one, or many of the loop controls. If multiple
552 loop controls are specified, all the specified controls will be applied and the first limit that is reached
553 causes the test to terminate. The loop controls that may be specified are 1 (Other), 2 (Continuous),
554 3 (Count), 4 (Timer), 5 (ErrorCount), 0x8000 (No Loop Control).

555 7.3.3.1 Other

556 The “Other” loop control is provided for vendor-specific loop controls. If this mode is specified, the
557 OtherLoopControlDescriptions should have at least one value.

558 NOTE If "Other" is specified, the implementing vendor should also extend the CIM_DiagnosticSettingData class to
559 include any loop control specification support needed.

560 **7.3.3.2 Continuous**

561 If this loop control is supported, the client may request that the test will execute continuously. The client
562 requests this mode by specifying 2 (Continuous) in the LoopControl property of the DiagnosticSettings
563 parameter (an embedded instance of CIM_DiagnosticSettingData) of the RunDiagnosticService()
564 method.

565 NOTE If a loop control of 2 (Continuous) is specified, the corresponding LoopControlParameter property of the
566 DiagnosticSettings parameter is ignored.

567 NOTE If a LoopControl of 0x8000 (No Loop Control) is specified, no other entry should be in the array property.
568 No Loop Control means the client cannot specify loop controls in the DiagnosticSettingData.

569 **7.3.3.3 Count**

570 If this loop control is supported, the client may request that the test will execute a specified number of
571 times with a single invocation of a test method. The client requests this mode by specifying 3 (Count) in
572 the LoopControl property and the number of loops desired in the LoopControlParameter property of the
573 DiagnosticSettings parameter (an embedded instance of CIM_DiagnosticSettingData) of the
574 RunDiagnosticService() method. The corresponding LoopControlParameter property specifies the count
575 in string format.

576 **7.3.3.4 Timer**

577 If this loop control is supported, the client may request that the test will execute for a specified number of
578 seconds and then terminate. The client requests this mode by specifying 4 (Timer) in the LoopControl
579 property and the number of seconds in the LoopControlParameter property of the DiagnosticSettings
580 parameter (an embedded instance of CIM_DiagnosticSettingData) of the RunDiagnosticService()
581 method. The corresponding LoopControlParameter property specifies the timer (in seconds) in string
582 format.

583 **7.3.3.5 ErrorCount**

584 If this loop control is supported, the client may request that the test will execute until the number of errors
585 that have occurred exceeds a specified ErrorCount. The client requests this mode by specifying 5
586 (ErrorCount) in the LoopControl property and the error count in the LoopControlParameter property of the
587 DiagnosticSettings parameter (an embedded instance of CIM_DiagnosticSettingData) of the
588 RunDiagnosticService() method. The corresponding LoopControlParameter property specifies the error
589 count in string format.

590 NOTE The ErrorCount only refers to device errors. It does not include processing errors or warnings.

591 **7.3.3.6 No Loop Control**

592 If this loop control is specified, the test shall have no loops. The client may not specify this mode; it is a
593 capability of the test. The implementation will ignore any loop control settings.

594 **7.3.4 CIM_DiagnosticServiceCapabilities.OtherSupportedLoopControlDescriptions**

595 The OtherSupportedLoopControlDescriptions provide additional information for SupportedLoopControl
596 when the corresponding value is set to 1 ("Other"). This property is intended for vendor-specific
597 extensions to the profile.

598 **7.3.5 CIM_DiagnosticServiceCapabilities.SupportedLogOptions**

599 This property identifies the log options supported by the DiagnosticTest. If any log options are supported,
600 they shall be published by using this property. Multiple entries may be provided in the
601 SupportedLogOptions. That is, a test may support none, one, or many of the log options. The options that
602 may be specified are 1 (Other), 2 (Subtests), 3 (Results), 4 (Actions), 5 (Warnings), 6 (Status), 7 (Device
603 Errors), 8 (Service Errors), 9 (Setting Data), 10 (Statistics), 11 (Hardware Configuration), 12 (Software
604 Configuration), 13 (References), 14 (Debug), and 0x8000 (No Log Options).

605 **7.3.5.1 Other**

606 The “Other” log option is provided for vendor-specific log options. If this option is specified, the
607 OtherLogOptionsDescriptions should have at least one value.

608 NOTE If “Other” is specified, the implementing vendor should also extend the CIM_DiagnosticSettingData class to
609 include any log option specification support needed.

610 **7.3.5.2 Subtests**

611 If this log option is supported, the client may request the test produce a summary report upon completion
612 of each subtest and each loop iteration. The client requests this option by specifying a value of 3 in the
613 LogOptions property of the DiagnosticSettings parameter (an embedded instance of
614 CIM_DiagnosticSettingData) of the RunDiagnosticService() method. The summary reports should state
615 whether the individual subtest or iteration passed or failed and list relevant error codes and respective
616 error counts.

617 **7.3.5.3 Results**

618 If this log option is supported, the client may request that the test log the results. The client requests this
619 option by specifying a value of 2 in the LogOptions property of the DiagnosticSettings parameter (an
620 embedded instance of CIM_DiagnosticSettingData) of the RunDiagnosticService() method. This option is
621 the most common value for reporting the test results.

622 NOTE This RecordType may also be specified for interim results from subtests.

623 **7.3.5.4 Actions**

624 If this log option is supported, the client may request that the test log corrective action and instructional
625 messages to guide service personnel. The client requests this option by specifying a value of 4 in the
626 LogOptions property of the DiagnosticSettings parameter (an embedded instance of
627 CIM_DiagnosticSettingData) of the RunDiagnosticService() method. For example, the test might present
628 a prioritized list of actions to perform to isolate a failure or correct a problem. When ordering steps or
629 prioritizing actions, a number should precede the text. For example, 1) Do this first, 2) Do this next, etc.

630 **7.3.5.5 Warnings**

631 If this log option is supported, the client may request that the test log warning messages. The client
632 requests this option by specifying a value of 5 in the LogOptions property of the DiagnosticSettings
633 parameter (an embedded instance of CIM_DiagnosticSettingData) of the RunDiagnosticService()
634 method. For example, log records for alerts that identify warnings (such as DIAG4, see 7.9.4) would be
635 logged.

636 **7.3.5.6 Status**

637 If this log option is supported, the client may request that the test log status messages. The client
638 requests this options by specifying a value of 6 in the LogOptions property of the DiagnosticSettings
639 parameter (an embedded instance of CIM_DiagnosticSettingData) of the RunDiagnosticService()
640 method. For example, the test might log status messages about state information for the driver, device, or
641 system.

642 **7.3.5.7 Device Errors**

643 If this log option is supported, the client may request that the test log errors related to the managed
644 element being tested. The client requests this option by specifying a value of 7 in the LogOptions property
645 of the DiagnosticSettings parameter (an embedded instance of CIM_DiagnosticSettingData) of the
646 RunDiagnosticService() method.

647 **7.3.5.8 Service Errors**

648 If this log option is supported, the client may request that the test log errors related to the test itself rather
649 than the element being tested. The client requests this option by specifying a value of 8 in the LogOptions
650 property of the DiagnosticSettings parameter (an embedded instance of CIM_DiagnosticSettingData) of
651 the RunDiagnosticService() method.

652 Support for this option means that the test logs errors related to the test itself rather than the element
653 being tested, such as log records associated with DIAG26 (see 7.9.20).

654 **7.3.5.9 Setting Data**

655 If this log option is supported, the client may request that the test log the property values of the
656 DiagnosticSettingData object that is used to configure the test. The client requests this option by
657 specifying a value of 9 in the LogOptions property of the DiagnosticSettings parameter (an embedded
658 instance of CIM_DiagnosticSettingData) of the RunDiagnosticService() method.

659 **7.3.5.10 Statistics**

660 If this log option is supported, the client may request that the test log statistical messages. The client
661 requests this option by specifying a value of 10 in the LogOptions property of the DiagnosticSettings
662 parameter (an embedded instance of CIM_DiagnosticSettingData) of the RunDiagnosticService()
663 method.

664 Support for this option means the client may request that the test log statistical messages, such as
665 packets sent per second.

666 **7.3.5.11 Hardware Configuration**

667 If this log option is supported, the client may request that the test log messages that contain information
668 about the hardware configuration as viewed by the test. The client requests this option by specifying a
669 value of 11 in the LogOptions property of the DiagnosticSettings parameter (an embedded instance of
670 CIM_DiagnosticSettingData) of the RunDiagnosticService() method. This information might include
671 vendor, version, FRU identification, and location information. The format and contents of this property are
672 element dependent.

673 **7.3.5.12 Software Configuration**

674 If this log option is supported, the client may request that the test log messages that contain information
675 about the software environment as viewed by the test. The client requests this option by specifying a
676 value of 12 in the LogOptions property of the DiagnosticSettings parameter (an embedded instance of
677 CIM_DiagnosticSettingData) of the RunDiagnosticService() method. This information might include the
678 name and version of all the critical software elements controlling the device under test. Each configuration
679 message should have the following common format: element name; element type; manufacturer name;
680 version.

681 **7.3.5.13 References**

682 If this log option is supported, the client may request that the test log the keys of an CIM object of interest.
683 The client requests this option by specifying a value of 13 in the LogOptions property of the
684 DiagnosticSettings parameter (an embedded instance of CIM_DiagnosticSettingData) of the

685 RunDiagnosticService() method. Typically this information should include the keys of the object under
686 test. However, it might also include the keys of the DiagnosticsLog.

687 **7.3.5.14 Debug**

688 If this log option is supported, the client may request that the test log debug messages. The client
689 requests this option by specifying a value of 14 in the LogOptions property of the DiagnosticSettings
690 parameter (an embedded instance of CIM_DiagnosticSettingData) of the RunDiagnosticService()
691 method. The debug messages would be vendor-specific messages to aid in debugging the test logic.

692 **7.3.5.15 No Log Messages**

693 If this log option is supported, the client may request that the test not log any messages. The client
694 requests this option by specifying a value of 0x8000 in the LogOptions property of the DiagnosticSettings
695 parameter (an embedded instance of CIM_DiagnosticSettingData) of the RunDiagnosticService()
696 method.

697 **7.3.6 CIM_DiagnosticServiceCapabilities.OtherSupportedLogOptionsDescriptions**

698 The OtherSupportedLogOptionsDescriptions provide additional information for SupportedLogOptions
699 when the corresponding value is set to 1 ("Other"). This option is intended for vendor-specific extensions
700 to the profile.

701 **7.3.7 CIM_DiagnosticServiceCapabilities.SupportedLogStorage**

702 This property identifies the log storage options supported by the DiagnosticTest. Multiple entries may be
703 provided in the SupportedLogStorage. That is, a test may support none, one, or many of the log storage
704 options. The options that may be specified are 1 (Other), 2 (DiagnosticLog), and 0x8000 (No Log
705 Storage).

706 **7.3.7.1 Other**

707 The "Other" log storage property is provided for vendor-specific log storage. If this property is specified,
708 the OtherLogStorageDescriptions should have at least one value.

709 NOTE If "Other" is specified, the implementing vendor should also extend the CIM_DiagnosticSettingData class to
710 include any log storage specification support needed.

711 **7.3.7.2 DiagnosticLog**

712 If this log storage is supported, the client may request that the test use a DiagnosticLog class for
713 aggregating diagnostic records. The client requests this option by specifying a value of 2 in the
714 LogStorage property of the DiagnosticSettings parameter (an embedded instance of
715 CIM_DiagnosticSettingData) of the RunDiagnosticService() method.

716 **7.3.7.3 No Log Storage**

717 If this log storage is specified, the client may not request any form of log storage. If anything is specified
718 in the DiagnosticSettings parameter, it will be ignored and a DIAG43 alert message will be issued (see
719 7.9.27).

720 **7.3.8 CIM_DiagnosticServiceCapabilities.OtherSupportedLogStorageDescriptions**

721 The OtherSupportedLogStorageDescriptions provide additional information for SupportedLogStorage
722 when the corresponding value is set to 1 ("Other"). This option is intended for vendor-specific extensions
723 to the profile.

724

725 **DEPRECATED**

726 **7.3.9 CIM_DiagnosticServiceCapabilities.SupportedExecutionControls**

727 NOTE CIM_DiagnosticServiceCapabilities.SupportedExecutionControls and
728 CIM_DiagnosticServiceCapabilities.OtherSupportedExecutionControlsDescriptions are being deprecated
729 in favor of CIM_DiagnosticServiceJobCapabilities.RequestedStatesSupported.

730 This option identifies the execution control options supported by the DiagnosticTest. Multiple entries may
731 be provided in the SupportedExecutionControls. That is, a test may support none, one, or many of the
732 execution controls. The options that may be specified are 1 (Other), 3 (Kill Job), 4 (Suspend Job), 5
733 (Terminate Job), 0x8000 (No Execution Controls).

734 **7.3.10 CIM_DiagnosticServiceCapabilities.OtherSupportedExecutionControlsDescription** 735 **s**

736 The OtherSupportedExecutionControlsDescriptions provide additional information for
737 SupportedExecutionControls when the corresponding value is set to 1 ("Other"). This option is intended
738 for vendor-specific extensions to the profile.

739 **DEPRECATED**

740

741 **7.4 CIM_DiagnosticSettingData**

742 This class defines specific diagnostic service parameters and execution instructions. To provide more
743 detailed settings for a type of test (that is, additional properties), subclassing is appropriate. This class
744 can be used in two different ways: 1) by the test to optionally publish its default settings; or 2) by the client
745 to optionally override the test default settings.

746 NOTE A CIM_DiagnosticSettingData object shall not contain any values that are unsupported by the diagnostic
747 service's CIM_DiagnosticServiceCapabilities object. For example, if
748 CIM_DiagnosticServiceCapabilities.SupportedLoopControl includes the value 5 (No Loop Control),
749 CIM_DiagnosticSettingData.LoopControl cannot include the value 3 (Count). Unsupported values shall be ignored by
750 the implementation.

751 **7.4.1 CIM_DiagnosticSettingData.HaltOnError**

752 When the default DiagnosticSettingData version of this property is TRUE, the test should halt after finding
753 the first error. If the implementation includes a DiagnosticServiceCapabilities instance for the test,
754 HaltOnError shall only be set to TRUE when DiagnosticServiceCapabilities.SupportedServiceModes
755 includes HaltOnError. If HaltOnError is not included in the
756 DiagnosticServiceCapabilities.SupportedServiceModes, the default DiagnosticSettingData.HaltOnError
757 shall be set to FALSE.

758 If a client sets HaltOnError to TRUE in the DiagnosticsSettings parameter for a RunDiagnosticService
759 method when the DiagnosticServiceCapabilities.SupportedServiceModes does not include HaltOnError,
760 the HaltOnError specification will be ignored. The unsupported setting parameter will result in a DIAG43
761 alert indication (see subclause 7.9.27) identifying that HaltOnError is not supported, if the client has
762 subscribed to the indication.

763 If HaltOnError is in effect, at the first device error, an alert message indicating the test was terminated
764 based on HaltOnError will be sent to any client subscribed to the indication. (See 7.9.8.)

765 **7.4.2 CIM_DiagnosticSettingData.QuickMode**

766 When the default DiagnosticSettingData version of this property is TRUE, the test should attempt to run in
767 an accelerated manner either by reducing the coverage or by reducing the number of tests performed. If

768 the implementation includes a DiagnosticServiceCapabilities instance for the test, QuickMode should only
769 be set to true when DiagnosticServiceCapabilities.SupportedServiceModes includes QuickMode.

770 If a client sets QuickMode to TRUE in the DiagnosticsSettings parameter for a RunDiagnosticService
771 method when the DiagnosticServiceCapabilities.SupportedServiceModes does not include QuickMode,
772 the QuickMode specification will be ignored. This conflict will result is a DIAG43 alert indication (see
773 subclause 7.9.27) identifying that QuickMode is not supported, if the client has subscribed to the
774 indication.

775 **7.4.3 CIM_DiagnosticSettingData.PercentOfTestCoverage**

776 This property requests the test to reduce test coverage to the specified percentage. If the implementation
777 includes a DiagnosticServiceCapabilities instance for the test, PercentOfTestCoverage should only be set
778 to true when DiagnosticServiceCapabilities.SupportedServiceModes includes PercentOfTestCoverage.

779 If a client sets PercentOfTestCoverage to anything other than NULL in the DiagnosticsSettings parameter
780 for a RunDiagnosticService method when the DiagnosticServiceCapabilities.SupportedServiceModes
781 does not include PercentOfTestCoverage, the PercentOfTestCoverage specification will be ignored. This
782 conflict will result is a DIAG43 alert indication (see subclause 7.9.27) identifying that
783 PercentOfTestCoverage is not supported, if the client has subscribed to the indication.

784 **7.4.4 CIM_DiagnosticSettingData.NonDestructive**

785 When the default DiagnosticSettingData version of this property is TRUE, the test should not run any
786 tests that would be destructive to the device or data on the device. If the implementation includes a
787 DiagnosticServiceCapabilities instance for the test, NonDestructive should only be set to TRUE when
788 DiagnosticServiceCapabilities.SupportedServiceModes includes NonDestructive.

789 If a client sets NonDestructive to TRUE in the DiagnosticsSettings parameter for a RunDiagnosticService
790 method when the DiagnosticServiceCapabilities.SupportedServiceModes does not include
791 NonDestructive, the test will be terminated without executing.

792 **7.4.5 CIM_DiagnosticSettingData.LoopControl and LoopControlParameter**

793 The LoopControl property is used in combination with the LoopControlParameter to set one or more loop
794 control mechanisms that limit the number of times that a test should be repeated.

795 With these properties, it is possible to loop a test (if supported) under control of a counter, timer, and
796 other loop terminating facilities. If the implementation includes a DiagnosticServiceCapabilities instance
797 for the test, LoopControl should only be set to a value contained in the
798 DiagnosticServiceCapabilities.SupportedLoopControl property (see subclause 7.3.3).

799 NOTE The No Loop Control option cannot be specified in the DiagnosticSettingData property. It is a capability of
800 the implementation, but cannot be requested.

801 **7.4.6 CIM_DiagnosticSettingData.ResultPersistence**

802 This property specifies how many seconds the log records should persist after service execution finishes.
803 If the implementation includes a DiagnosticServiceCapabilities instance for the test, ResultPersistence
804 should be set when DiagnosticServiceCapabilities.SupportedServiceModes includes ResultPersistence. If
805 ResultPersistence is not specified in the DiagnosticServiceCapabilities.SupportedServiceModes, but is
806 specified in the default DiagnosticSettingData, the results will be retained for the default
807 ResultPersistence value.

808 ResultPersistence is specified in seconds. If it is set to zero (0), the provider need not persist the diagnostic result.
809 The diagnostic information is only available while the diagnostic is executing or at its conclusion.

810 If ResultPersistence is set to 0xFFFFFFFF, the provider shall persist results forever. In this case, the client bears the
811 responsibility for deleting them. An implementation might not support the 0xFFFFFFFF value even though it claims
812 support for ResultPersistence in its DiagnosticServiceCapabilities.SupportedServiceModes. If the client
813 request to persist results forever is rejected, the client may specify any other value and the
814 implementation shall support that time period.

815 NOTE Results (e.g., DiagnosticLog information) are independent of the job that creates the results. The life of the
816 job is controlled by the TimeBeforeRemoval property of the ConcreteJob. The life of the DiagnosticLog information is
817 controlled by ResultPersistence. One may be deleted before the other.

818 **7.4.7 CIM_DiagnosticSettingData.LogOptions**

819 This property specifies the types of data that should be logged by the diagnostic test.

820 This property supports specification of the nature of data being logged by the test through the addition of
821 the LogOptions enumeration. If the implementation includes a DiagnosticServiceCapabilities instance for
822 the test, LogOptions should only be set to a value contained in the
823 DiagnosticServiceCapabilities.SupportedLogOptions property (see subclause 7.3.5).

824 **7.4.8 CIM_DiagnosticSettingData.LogStorage**

825 This property specifies the logging mechanism to store the diagnostic results. If the implementation
826 includes a DiagnosticServiceCapabilities instance for the test, LogStorage should only be set to a value
827 contained in the DiagnosticServiceCapabilities.SupportedLogStorage property (see subclause 7.3.7).

828 NOTE The No Log Storage option cannot be specified in DiagnosticSettingData.

829 **7.4.9 CIM_DiagnosticSettingData.VerbosityLevel**

830 This property specifies the desired volume or detail logged by a diagnostic test. The possible values
831 include Minimum, Standard, and Full. The exact meaning of each of these are vendor specific. The
832 definitions in this subclause are guidelines.

833 The VerbosityLevel property is an array property that is correlated with the LogOptions property. That is,
834 VerbosityLevel can be set for each log option specified in the LogOptions setting property.

835 In the default CIM_DiagnosticSettingData, the provider would identify the default VerbosityLevel for each
836 of the log options that it supports.

837 In the DiagnosticSettings parameter (an embedded instance of CIM_DiagnosticSettingData) of the
838 RunDiagnosticService method, the client would specify the verbosity level desired for each of the log
839 options.

840 **7.4.9.1 Minimum**

841 This value would be specified if the least amount of information is desired.

842 **7.4.9.2 Standard**

843 This level is the standard level of messaging provided by the test. This value would be specified to get the
844 default level of logging.

845 **7.4.9.3 Full**

846 This value would be specified when all information, regardless of size, is desired.

847 **7.4.10 Default setting**

848 The default settings for a diagnostic service are obtained by using the CIM_ElementSettingData
849 association to an instance of (a subclass of) CIM_DiagnosticSettingData where the IsDefault property has
850 the value of TRUE.

851 **7.4.11 Client override**

852 A client can choose to accept the default settings (published or not) or override the default settings by
853 creating a CIM_DiagnosticSettingData object based upon the settings that an implementation indicates
854 are supported in its CIM_DiagnosticServiceCapabilities object.

855 If a client chooses to accept the default settings (published or not), the DiagnosticSettings argument to
856 the RunDiagnosticService() method of DiagnosticTest should be set to NULL or an empty string.

857 If a client chooses to override the default settings, the DiagnosticSettings argument to the
858 RunDiagnosticService() method of DiagnosticTest is set to an encoded form of the
859 CIM_DiagnosticSettingData object.

860 Note that the CIM_DiagnosticSettingData subclass may have extensions. If the client is aware of the
861 extensions, these may be modified as well. If the client is unaware, the default values should be used.

862 **7.5 CIM_DiagnosticLog**

863 All diagnostic result messages shall be represented by instances of CIM_DiagnosticRecord subclasses.
864 Moreover, those records shall be aggregated to an instance of CIM_DiagnosticLog. Each invocation of
865 the RunDiagnosticService method of DiagnosticTest shall instantiate a new CIM_DiagnosticLog object. A
866 diagnostic service may also implement other additional logging mechanisms. Any other implemented
867 logging mechanism shall be indicated in the LogStorage property of the published capabilities.

868 **7.5.1 Logging results**

869 The methods to record the results of running a diagnostic service are specified by the LogOptions and
870 LogStorage properties of the CIM_DiagnosticSettingData class. Use LogOptions to specify *what* to log
871 and LogStorage to specify *where* to log it. The MOF file describes these properties in some detail, but it is
872 useful to emphasize the mandatory mechanism here.

873 *Diagnostic Records aggregated to the Diagnostic Log* is mandatory for several reasons:

- 874 • The heterogeneous nature of the log entries more easily fits into a self-describing record
875 paradigm.
- 876 • Keyed records are easier to manage and retrieve.

877 **7.6 CIM_DiagnosticRecord**

878 CIM_DiagnosticRecord has two subclasses: CIM_DiagnosticServiceRecord and
879 CIM_DiagnosticSettingDataRecord. CIM_DiagnosticServiceRecord has a single subclass:
880 CIM_DiagnosticCompletionRecord.

881 CIM_DiagnosticServiceRecord is structured to hold the information that is generated while a particular
882 service is running. One or more CIM_DiagnosticServiceRecord objects may be created during a single
883 execution of a test.

884 CIM_DiagnosticSettingDataRecord is structured to hold the attributes of the setting object that was used
885 as the DiagnosticSettings parameter to the RunDiagnosticService() method. The record that gets written
886 to the log is the "effective" DiagnosticSettingData that includes default and overridden values. At most, a
887 single CIM_DiagnosticSettingDataRecord may be created during a single execution of a test.

888 CIM_DiagnosticCompletionRecord is structured to hold the information that is generated as a result of
889 running the particular service. A single CIM_DiagnosticCompletionDataRecord shall be created during a
890 single execution of a test.

891 **7.6.1 CIM_DiagnosticRecord.ExpirationDate**

892 After a diagnostic service produces results, the result objects need to persist for a minimum amount of
893 time to allow diagnostic CIM clients to capture what the application needs. When the data has been
894 captured, the containing objects need to be deleted in a timely manner.

895 CIM_DiagnosticSettingData.ResultPersistence shall be used by the client to specify to the diagnostic
896 service implementation how long the results generated by that service shall persist. A value shall be

897 chosen that allows the minimum time needed by the client to record the data. When the timeout value has
898 been reached, the implementation shall delete the data objects that contain the results.

899 The value of CIM_DiagnosticRecord.ExpirationDate shall be calculated by the implementation to account
900 for the persistence setting value, time zone, and other applicable factors. When this expiration value has
901 been reached, the record is eligible for immediate deletion by the implementation. It is the
902 implementation's responsibility to manage the logs to prevent accumulation of expired records.

903 A ResultPersistence value of 0 (zero) indicates that the result does not need to persist; the
904 ExpirationDate is set to the current date and time. A ResultPersistence value of 0xFFFFFFFF indicates
905 that the result shall persist until it is explicitly deleted by a client DeleteInstance or ClearLog call; the
906 ExpirationDate is set to NULL, indicating no expiration date.

907 **7.6.2 CIM_DiagnosticRecord.InstanceID**

908 To simplify the retrieval of test data for a specific test execution, the value of InstanceID for
909 CIM_ConcreteJob is closely related to the InstanceID for the subclasses of CIM_DiagnosticRecord.

910 CIM_DiagnosticRecord.InstanceID should be constructed by using the following preferred algorithm:

911 <ConcreteJob.InstanceID>:<n>

912 <ConcreteJob.InstanceID> is <OrgID>:<LocalID> as described in CIM_ConcreteJob, and <n> is an
913 increment value that provides uniqueness. <n> should be set to 0 for the first record created by the test
914 during this job, and incremented for each subsequent record created by the test during this job. Each new
915 test execution can reset the <n> to 0.

916 **7.6.3 CIM_DiagnosticRecord.RecordType**

917 The RecordType property of DiagnosticRecords correlates with the
918 CIM_DiagnosticSettingData.LogOptions property. Each DiagnosticRecord in the log identifies the
919 RecordType for the LogOptions value specified in the DiagnosticSettingData. If a Log Option is not
920 included in the DiagnosticSettingData, the DiagnosticRecords that would have contained that
921 RecordType shall not be logged.

922 The RecordType also identifies which DiagnosticProperties are populated.

923 **7.7 CIM_ServiceComponent**

924 CIM_ServiceComponent is the means by which clients discover any individual tests that are also subtests
925 within a packaging test. This association does not imply any order, number, or method of subtest
926 execution, nor that all subtests executed within a packaging test shall be individual tests, nor even that all
927 the subtests would be executed for any specific execution of the packaging test.

928 The packaging test shall ensure that the values in CIM_DiagnosticTest.Characteristics of the packaging
929 test are consistent with the values in CIM_DiagnosticTest.Characteristics of the subtests unless the
930 packaging test can execute the subtest such that it does not have those characteristics. For example, if a
931 subtest sets the values of 4 (Is Destructive) or 3 (Is Interactive), the packaging test values in
932 CIM_DiagnosticTest.Characteristics should reflect those same characteristics, unless the packaging test
933 can execute the subtest so that it is not destructive or interactive.

934 **7.8 Diagnostics Profile Indications support**

935 The Diagnostics Profile constrains certain elements in its support for the DMTF Indications Profile. This
936 subclause identifies those constraints.

937 **7.8.1 CIM_IndicationFilter (StaticIndicationFilter)**

938 The Diagnostics Profile constrains some of the properties of the StaticIndicationFilter version of the
939 CIM_IndicationFilter class and makes the class mandatory. The class is mandatory because some of the
940 alert indication filters are mandatory and the Diagnostics Profile requires that static versions of mandatory
941 indication filters be populated.

942 **7.8.1.1 CIM_IndicationFilter.Name**

943 The Diagnostics Profile constrains names of the profile defined alert indication filters as prescribed by
944 [DSP1054](#). The names for the indication filters are identified in the entries for the indications in Table 28.
945 The Name shall be formatted as defined by the following ABNF rule:

946 "DMTF:Diagnostics:" MessageID

947 The MessageID shall have the same value of the MessageID in the Query for the filter.

948 **7.8.1.2 CIM_IndicationFilter.Query**

949 The Diagnostics Profile constrains the Query properties of the profile defined alert indication filters as
950 prescribed by [DSP1054](#). The Query properties for the indication filters are identified in the entries for the
951 indications in Table 28.

952 **7.8.1.3 CIM_IndicationFilter.QueryLanguage**

953 The Diagnostics Profile constrains the QueryLanguage properties of the profile defined alert indication
954 filters as prescribed by [DSP1054](#). The QueryLanguage properties for the indication filters are identified in
955 the entries for the indications in Table 28.

956 **7.8.2 CIM_FilterCollection (ProfileSpecificFilterCollection)**

957 The Diagnostics Profile constrains the CollectionName property of the ProfileSpecificFilterCollection
958 version of the CIM_FilterCollection class.

959 **7.8.2.1 CIM_FilterCollection.CollectionName**

960 The Diagnostics Profile constrains CollectionName of the profile defined ProfileSpecificFilterCollection
961 filter collection as prescribed by [DSP1054](#). The CollectionName for the filter collection shall be formatted
962 as defined by the following ABNF rule:

963 "DMTF:Diagnostics:ProfileSpecifiedAlertIndicationFilterCollection"

964 **7.8.3 CIM_MemberOfCollection (IndicationFilterInFilterCollection)**

965 The Diagnostics Profile constrains the properties of the IndicationFilterInFilterCollection version of the
966 CIM_MemberOfCollection class.

967 **7.8.3.1 CIM_MemberOfCollection.Collection**

968 The Diagnostics Profile constrains the Collection property to be the reference to the
969 ProfileSpecificFilterCollection filter collection.

970 **7.8.3.2 CIM_MemberOfCollection.Member**

971 The Diagnostics Profile constrains the Member property to be a reference to one of the profile defined
972 alert indication filters.

973 **7.8.4 CIM_OwningCollectionElement (IndicationServiceOfFilterCollection)**

974 The Diagnostics Profile constrains the OwnedElement property of the IndicationServiceOfFilterCollection
975 version of the CIM_OwningCollectionElement class.

976 **7.8.4.1 CIM_OwningCollectionElement.OwnedElement**

977 The Diagnostics Profile constrains OwnedElement property to be the reference to the
978 ProfileSpecificFilterCollection filter collection.

979 **7.9 Diagnostics alert indications and standard messages**

980 **7.9.1 DIAG0 – The test passed.**

981 The test executed with no errors or warnings.

982 This alert would be sent if the inputs were accepted, defaulted, or reset and the test executed to
983 completion (as defined by the inputs) and the test reported no errors or warnings.

984 This indication is the last alert indication that would be sent in a successful test execution. Any indications
985 that precede it would be informational messages.

986 The variables in this message are:

- 987 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
988 property of the DiagnosticTest instance.
- 989 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
990 that was specified.

991 This could be one of the following:

- 992 – The Object Path of the element
- 993 – The ElementName of the element
- 994 – A unique user friendly name not in the model (such as, asset name)

995 The Element Moniker can be any of these, but whichever one is used shall be used consistently
996 for all managed elements of the same type within the scoping profile (such as, all disk drives in
997 a system).

- 998 • Log Object Path – Identifies the Object Path of the CIM_DiagnosticLog instance.

999 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
1000 “Successful Completion”.

1001 With this alert, the PerceivedSeverity shall have the value 2 (Information).

1002 **7.9.2 DIAG1 – The reason for the test failure is unknown**

1003 The test failed to execute for unknown reasons.

1004 This alert would be sent if the test failed to execute for any one of a number of reasons. A client should
1005 refer to other alert indications that may have been sent to determine what (if anything) can be done.

1006 The variables in this message are:

- 1007 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1008 property of the DiagnosticTest instance.
- 1009 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1010 that was specified.

- 1011 This could be one of the following:
- 1012 – The Object Path of the element
- 1013 – The ElementName of the element
- 1014 – A unique user friendly name not in the model (such as, asset name)
- 1015 The Element Moniker can be any of these, but whichever one is used shall be used consistently
- 1016 for all managed elements of the same type within the scoping profile (such as, all disk drives in
- 1017 a system).
- 1018 • Log Object Path – Identifies the Object Path of the CIM_DiagnosticLog instance.
- 1019 • AlertType – Identifies the AlertType for this alert indication (such as, Processing Error or Device
- 1020 Alert).
- 1021 With this alert the AlertType shall have the value 4 (Processing Error) or 5 (Device Alert). If the test failed
- 1022 before the actual test was started, the AlertType shall have the value 4 (Processing Error). If the test had
- 1023 started and then failed, the AlertType should have the value 5 (Device Alert).
- 1024 With this alert, the PerceivedSeverity shall have the value 0 (Unknown), 4 (Minor), 5 (Major), 6 (Critical),
- 1025 or 7 (Fatal/NonRecoverable). If the severity is unknown, 0 (unknown) should be specified. If this is a
- 1026 processing error (see above), the PerceivedSeverity should be coded as 4 (Minor) or 5 (Major). If this is a
- 1027 device alert (see above), this may be 4 (Minor), 5 (Major), 6 (Critical) or 7 (Fatal/NonRecoverable). It
- 1028 should only be 7 (Fatal/NonRecoverable) if recovery cannot be done.
- 1029 **7.9.3 DIAG3 – The device test failed**
- 1030 The test ran, but the element under test reported device alert errors.
- 1031 This alert would only be sent if the element under test reported errors. This would be the last Alert
- 1032 Indication and device error alerts should have preceded this alert.
- 1033 It would NOT be sent if
- 1034 • there were errors in processing the RunDiagnosticsService parameters, or
- 1035 • the element under test only issued warnings.
- 1036 The variables in this message are:
- 1037 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
- 1038 property of the DiagnosticTest instance.
- 1039 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
- 1040 that was specified.
- 1041 This could be one of the following:
- 1042 – The Object Path of the element
- 1043 – The ElementName of the element
- 1044 – A unique user friendly name not in the model (such as, asset name)
- 1045 The Element Moniker can be any of these, but whichever one is used shall be used consistently
- 1046 for all managed elements of the same type within the scoping profile (such as, all disk drives in
- 1047 a system).
- 1048 • Log Object Path – Identifies the Object Path of the CIM_DiagnosticLog instance.
- 1049 With this alert, the AlertType shall have the value 5 (Device Alert).
- 1050 With this alert, the PerceivedSeverity shall have the value 4 (Minor), 5 (Major), 6 (Critical), or 7
- 1051 (Fatal/NonRecoverable).

1052 **7.9.4 DIAG4 – The test completed with warnings**

1053 The test ran, but the element under test reported warnings.

1054 This alert would be sent if the test ran to completion with no errors, but reported warnings. This would be
1055 the last alert indication sent for the test run. Informational and warning messages may have preceded this
1056 message.

1057 The variables in this message are:

- 1058 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1059 property of the DiagnosticTest instance.
- 1060 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1061 that was specified.

1062 This could be one of the following:

- 1063 – The Object Path of the element
- 1064 – The ElementName of the element
- 1065 – A unique user friendly name not in the model (such as, asset name)

1066 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1067 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1068 a system).

- 1069 • Log Object Path – Identifies the Object Path of the CIM_DiagnosticLog instance.

1070 With this alert, the AlertType shall have the value 4 (Processing Error) or 5 (Device Alert). Processing
1071 Error means input was given but ignored or input was reset by the test job. Device Alert means the device
1072 reported the warning.

1073 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1074 **7.9.5 DIAG5 – The requested test is not supported**

1075 The test as requested in the RunDiagnosticService extrinsic method is not supported on the element
1076 specified.

1077 This alert would be sent if the instance of DiagnosticTest (and the test that it represents) is not supported
1078 on the ManagedElement specified by the RunDiagnosticService extrinsic method. The test may well be
1079 supported on other elements, but not on the element specified in the request.

1080 The variables in this message are:

- 1081 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1082 property of the DiagnosticTest instance.
- 1083 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1084 that was specified.

1085 This could be one of the following:

- 1086 – The Object Path of the element
- 1087 – The ElementName of the element
- 1088 – A unique user friendly name not in the model (such as, asset name)

1089 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1090 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1091 a system).

- 1092 • Log Object Path – Identifies the Object Path of the CIM_DiagnosticLog instance.

1093 With this alert, the AlertType shall have the value 5 (Device Alert). The device does not support the test.

1094 With this alert, the PerceivedSeverity shall have the value 3 (Warning). There is no device problem, but
1095 the test cannot be run on the specified element.

1096 **7.9.6 DIAG6 – The required test setup steps have not been performed**

1097 The test did not run because the proper set up steps were not done to support the test.

1098 This alert would be sent if in processing the request, the test detected that certain conditions are not
1099 present to execute the test. For example, a setup file is missing or the element in question is disabled or
1100 the device is not connected. This alert will be followed by a “test did not start” or “test aborted” test
1101 completion status alert (see 7.9.28 and 7.9.29).

1102 The variables in this message are:

1103 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1104 property of the DiagnosticTest instance.

1105 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1106 that was specified.

1107 This could be one of the following:

- 1108 – The Object Path of the element
- 1109 – The ElementName of the element
- 1110 – A unique user friendly name not in the model (such as, asset name)

1111 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1112 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1113 a system).

1114 With this alert, the AlertType shall have the value 4 (Processing Error). That is, the test was not run
1115 because the proper set up has not been done.

1116 With this alert, the PerceivedSeverity shall have the value 3 (Warning). That is, there is no real error with
1117 the element under test, just a setup error. The client needs to ensure that the proper setup is done before
1118 running the test again.

1119 **7.9.7 DIAG7 – The test ran but HaltOnError is not supported**

1120 The test ran and found one or more errors, but the test did not halt on the first error because HaltOnError
1121 is not supported by the test on the specified element.

1122 This alert would be sent if the DiagnosticSettings parameter of the RunDiagnosticService method
1123 included HaltOnError=TRUE, but the device did not support HaltOnError.

1124 The variables in this message are:

1125 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1126 property of the DiagnosticTest instance.

1127 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1128 that was specified.

1129 This could be one of the following:

- 1130 – The Object Path of the element
- 1131 – The ElementName of the element
- 1132 – A unique user friendly name not in the model (such as, asset name)

1133 The Element Moniker can be any of these, but whichever one is used shall be used consistently
 1134 for all managed elements of the same type within the scoping profile (such as, all disk drives in
 1135 a system).

1136 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
 1137 Parameter Ignored.

1138 With this alert, the PerceivedSeverity shall have the value 3 (Warning). The Alert warns the client that the
 1139 test was not run with HaltOnError in effect.

1140 **7.9.8 DIAG8 – The test halted due to an error**

1141 The test ran until it found a Device Error and was terminated because the DiagnosticSettings parameter
 1142 of the RunDiagnosticService method called for HaltOnError.

1143 This alert would be sent if the client set HaltOnError to TRUE and the test encountered a Device Error.
 1144 The test does not run to completion, but it is terminated. The resulting JobState for the ConcreteJob is 8
 1145 (Terminated), just as though the client had issued a RequestedStateChange requesting termination.

1146 To determine the error that caused the test to be halted, see prior (error) alert indications or see the
 1147 DiagnosticLog records.

1148 The variables in this message are:

- 1149 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
 1150 property of the DiagnosticTest instance.
- 1151 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
 1152 that was specified.

1153 This could be one of the following:

- 1154 – The Object Path of the element
- 1155 – The ElementName of the element
- 1156 – A unique user friendly name not in the model (such as, asset name)

1157 The Element Moniker can be any of these, but whichever one is used shall be used consistently
 1158 for all managed elements of the same type within the scoping profile (such as, all disk drives in
 1159 a system).

1160 With this alert, the AlertType shall have the value 5 (Device Alert).

1161 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1162 **7.9.9 DIAG10 – QuickMode is not supported**

1163 The test ran but QuickMode is not supported.

1164 This alert would be sent if the client requested QuickMode=TRUE in the DiagnosticSettings parameter of
 1165 the RunDiagnosticService method, but QuickMode is not supported for the test or the element under test.
 1166 The QuickMode parameter applies to the test invoked by RunDiagnosticService (and may or may not
 1167 apply to lower level tests).

1168 The variables in this message are:

- 1169 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
 1170 property of the DiagnosticTest instance.
- 1171 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
 1172 that was specified.

1173 This could be one of the following:

- 1174 – The Object Path of the element
1175 – The ElementName of the element
1176 – A unique user friendly name not in the model (such as, asset name)
- 1177 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1178 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1179 a system).
- 1180 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
1181 Parameter Not Supported.
- 1182 With this alert, the PerceivedSeverity shall have the value 3 (Warning). The alert warns the client that the
1183 test was not run in QuickMode.
- 1184 **7.9.10 DIAG11 – Requested LoopControl type not supported**
- 1185 The test may or may not have run, but a LoopControl specified in the DiagnosticSettings parameter of the
1186 RunDiagnosticService method was not supported. Another test completion status alert indicates whether
1187 or not the test was run.
- 1188 This alert would be sent if the request asked for a LoopControl that was not supported by the test or the
1189 element under test.
- 1190 NOTE If multiple LoopControl types were not supported, multiple alert messages will be sent.
- 1191 The variables in this message are:
- 1192 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1193 property of the DiagnosticTest instance.
 - 1194 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1195 that was specified.
- 1196 This could be one of the following:
- 1197 – The Object Path of the element
 - 1198 – The ElementName of the element
 - 1199 – A unique user friendly name not in the model (such as, asset name)
- 1200 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1201 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1202 a system).
- 1203 • LoopControl – Identifies the LoopControl in the DiagnosticSettings parameter of the
1204 RunDiagnosticService method that was not supported.
- 1205 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to Loop
1206 Control Type Not Supported.
- 1207 With this alert, the PerceivedSeverity shall have the value 3 (Warning).
- 1208 **7.9.11 DIAG13 – Logging could not be started**
- 1209 The test ran, but the logging requested could not be started.
- 1210 This alert would be sent if a client requested some type of logging, but logging could not be started. If
1211 multiple logs could not be started, the client may receive multiple versions of this message. This alert
1212 would be sent as soon as the problem is discovered (before or as the test is running). Clients would have
1213 the opportunity to kill or terminate the test.
- 1214 The variables in this message are:

1215 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1216 property of the DiagnosticTest instance.

1217 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1218 that was specified.

1219 This could be one of the following:

- 1220 – The Object Path of the element
- 1221 – The ElementName of the element
- 1222 – A unique user friendly name not in the model (such as, asset name)

1223 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1224 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1225 a system).

1226 • Log Storage – Identifies the type of log storage the client requested.

1227 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType would be Log Not
1228 Started.

1229 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1230 **7.9.12 DIAG14 – Logging errors occurred**

1231 The test ran, but logging errors (such as, errors writing the log) occurred.

1232 This alert would be sent if the test ran and logging errors occurred in one of the logs specified by the
1233 DiagnosticSetting parameter of the RunDiagnosticService method request. If more than one log storage
1234 experiences errors, the multiple alerts will be sent. This message would be sent when the first error
1235 writing to the log is encountered.

1236 The variables in this message are:

1237 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1238 property of the DiagnosticTest instance.

1239 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1240 that was specified.

1241 This could be one of the following:

- 1242 – The Object Path of the element
- 1243 – The ElementName of the element
- 1244 – A unique user friendly name not in the model (such as, asset name)

1245 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1246 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1247 a system).

1248 • Log Storage – Identifies the log storage that experienced the errors.

1249 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to Log
1250 Errors Occurred.

1251 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1252 **7.9.13 DIAG15 – LogStorage type not supported**

1253 The test ran, but a LogStorage request was not supported.

- 1254 This alert would be sent if the client requested one or more log storage types, but one of them is not
1255 supported by the implementation.
- 1256 NOTE If multiple log storage types are not supported, multiple DIAG15 alerts would be sent. DIAG15 alerts do not
1257 report a mismatch between the setting and capabilities. That situation is handled by a separate alert (see 7.9.30,
1258 DIAG46 – LogStorage mismatch with capabilities).
- 1259 The variables in this message are:
- 1260
- Log Storage – Identifies the log storage that was requested.
- 1261
- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1262 property of the DiagnosticTest instance.
- 1263
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1264 that was specified.
- 1265 This could be one of the following:
- 1266
- The Object Path of the element
 - 1267 – The ElementName of the element
 - 1268 – A unique user friendly name not in the model (such as, asset name)
- 1269 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1270 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1271 a system).
- 1272 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to Log
1273 Storage Not Supported.
- 1274 With this alert, the PerceivedSeverity shall have the value 3 (Warning).
- 1275 **7.9.14 DIAG16 – LoopControl Parameter invalid**
- 1276 The test ran, but a LoopControlParameter supplied in the DiagnosticSetting parameter of the
1277 RunDiagnosticService method was invalid and ignored.
- 1278 This alert would be sent if a LoopControlParameter provided on the method is invalid. An invalid
1279 LoopControlParameter could be:
- 1280
- A string that could not be converted to a number or datetime datatype
 - 1281 • A string that converts to a number or datetime, but is not supported
- 1282 If multiple LoopControlParameters are invalid, multiple alert messages would be sent, one for each invalid
1283 LoopControlParameter.
- 1284 The variables in this message are:
- 1285
- Loop Control Parameter – Identifies the LoopControlParameter value supplied with the
1286 DiagnosticSetting parameter on the RunDiagnosticService request.
- 1287
- Loop Control – Identifies the LoopControl that was supplied with the DiagnosticSetting
1288 parameter on the RunDiagnosticService request for interpreting the LoopControlParameter
- 1289
- Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1290 property of the DiagnosticTest instance.
- 1291
- Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1292 that was specified.
- 1293 This could be one of the following:
- 1294
- The Object Path of the element
 - 1295 – The ElementName of the element

- 1296 – A unique user friendly name not in the model (such as, asset name)
- 1297 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1298 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1299 a system).
- 1300 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
1301 Parameter Ignored.
- 1302 With this alert, the PerceivedSeverity shall have the value 3 (Warning).
- 1303 **7.9.15 DIAG17 – VerbosityLevel not supported**
- 1304 The test ran, but the VerbosityLevel requested by the DiagnosticSetting parameter was not supported.
- 1305 This alert would be sent if the client requested a VerbosityLevel in the DiagnosticSetting parameter of the
1306 RunDiagnosticService method, but that VerbosityLevel is not supported. The default VerbosityLevel was
1307 used instead.
- 1308 The variables in this message are:
- 1309 • Verbosity Level Specified – Identifies the VerbosityLevel value supplied with the
1310 DiagnosticSetting parameter on the RunDiagnosticService request.
 - 1311 • Verbosity Level Used – Identifies the VerbosityLevel value used on the RunDiagnosticService
1312 request.
 - 1313 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1314 property of the DiagnosticTest instance.
 - 1315 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1316 that was specified.
- 1317 This could be one of the following:
- 1318 – The Object Path of the element
 - 1319 – The ElementName of the element
 - 1320 – A unique user friendly name not in the model (such as, asset name)
- 1321 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1322 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1323 a system).
- 1324 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
1325 Parameter Ignored.
- 1326 With this alert, the PerceivedSeverity shall have the value 3 (Warning).
- 1327 **7.9.16 DIAG18 – PercentOfTestCoverage level was not completed**
- 1328 The test ran, but the PercentOfTestCoverage level that was requested in the DiagnosticSetting parameter
1329 of the RunDiagnosticService method was not completed.
- 1330 This alert would be sent if the client requested a PercentOfTestCoverage level in the DiagnosticSetting
1331 parameter of the RunDiagnosticService method but the percentage was not achieved.
- 1332 The variables in this message are:
- 1333 • PercentRequested – Identifies the percent requested in the DiagnosticSetting parameter.
 - 1334 • PercentCompleted – Identifies the percent completed.

1335 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1336 property of the DiagnosticTest instance.

1337 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1338 that was specified.

1339 This could be one of the following:

- 1340 – The Object Path of the element
- 1341 – The ElementName of the element
- 1342 – A unique user friendly name not in the model (such as, asset name)

1343 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1344 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1345 a system).

1346 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
1347 Parameter ignored.

1348 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1349 **7.9.17 DIAG22 – ErrorCount exceeded**

1350 The test ran, but the ErrorCount specified in the LoopControlParameter of the DiagnosticSetting was
1351 exceeded and the test terminated.

1352 This alert would be sent if the client specified ErrorCount as one of the loop controls in the
1353 DiagnosticSetting the client supplied on the RunDiagnosticService method and the error count (as
1354 specified in the LoopControlParameter) has been achieved.

1355 The variables in this message are:

1356 • LoopControl Error Count – Identifies the LoopControlParameter requested (the count that was
1357 exceeded).

1358 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1359 property of the DiagnosticTest instance.

1360 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1361 that was specified.

1362 This could be one of the following:

- 1363 – The Object Path of the element
- 1364 – The ElementName of the element
- 1365 – A unique user friendly name not in the model (such as, asset name)

1366 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1367 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1368 a system).

1369 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to Loop
1370 Control Reached.

1371 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1372 **7.9.18 DIAG23 – LoopControl limit reached as configured by the client**

1373 The test ran, but the Count or Error Count specified in the LoopControlParameter of the DiagnosticSetting
1374 was reached and the test terminated.

1375 This alert would be sent if the client specified Count or Error Count as one of the loop controls in the
 1376 DiagnosticSetting that was supplied on the RunDiagnosticService method and the loop count (as
 1377 specified in the LoopControlParameter) has been achieved. If multiple LoopControl limits are reached,
 1378 there would be multiple messages.

1379 The variables in this message are:

- 1380 • Loop Control – Identifies which LoopControl limit was reached.
- 1381 • LoopControl Parameter Value – Identifies the LoopControlParameter requested (the count that
 1382 was reached).
- 1383 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
 1384 property of the DiagnosticTest instance.
- 1385 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
 1386 that was specified.

1387 This could be one of the following:

- 1388 – The Object Path of the element
- 1389 – The ElementName of the element
- 1390 – A unique user friendly name not in the model (such as, asset name)

1391 The Element Moniker can be any of these, but whichever one is used shall be used consistently
 1392 for all managed elements of the same type within the scoping profile (such as, all disk drives in
 1393 a system).

1394 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to Loop
 1395 Control Reached.

1396 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1397 **7.9.19 DIAG24 – LoopControl Timeout limit reached as configured by the client**

1398 The test ran, but the Timer specified in the LoopControlParameter of the DiagnosticSetting was reached
 1399 and the test terminated.

1400 This alert would be sent if the client specified Timer as one of the loop controls in the DiagnosticSetting
 1401 that was supplied on the RunDiagnosticService method and the loop time (as specified in the
 1402 LoopControlParameter) has been achieved.

1403 The variables in this message are:

- 1404 • LoopControl Parameter Value – Identifies the LoopControlParameter requested (the timer that
 1405 was reached).
- 1406 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
 1407 property of the DiagnosticTest instance.
- 1408 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
 1409 that was specified.

1410 This could be one of the following:

- 1411 – The Object Path of the element
- 1412 – The ElementName of the element
- 1413 – A unique user friendly name not in the model (such as, asset name)

1414 The Element Moniker can be any of these, but whichever one is used shall be used consistently
 1415 for all managed elements of the same type within the scoping profile (such as, all disk drives in
 1416 a system).

1417 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to Loop
1418 Control Reached.

1419 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1420 **7.9.20 DIAG26 – Test cannot be run with NonDestructive set to true**

1421 The test was not run because the client requested NonDestructive=TRUE in the DiagnosticSetting
1422 parameter of the RunDiagnosticService method and this function is not supported for the test or the
1423 element under test.

1424 This alert would be sent if the client requested a NonDestructive execution, but the implementation does
1425 not support this for the test or the element under test.

1426 NOTE This message would not be sent when the NonDesctructive value conflicts with the SupportedServiceModes
1427 property of the DiagnosticServiceCapabilities (see 7.9.26 for the message for a mismatch with capabilities). DIAG26
1428 would be sent if the optional DiagnosticServiceCapabilities was not implemented or the capabilities was implemented
1429 and the SupportedServiceModes include NonDestructive, but not for the element under test.

1430 The variables in this message are:

- 1431 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1432 property of the DiagnosticTest instance.
- 1433 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1434 that was specified.

1435 This could be one of the following:

- 1436 – The Object Path of the element
- 1437 – The ElementName of the element
- 1438 – A unique user friendly name not in the model (such as, asset name)

1439 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1440 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1441 a system).

1442 With this alert, the AlertType shall have the value 4 (Processing Error).

1443 With this alert, the PerceivedSeverity shall have the value 5 (Major).

1444 **7.9.21 DIAG27 – Capability to set LoopControl not supported**

1445 The test ran, but a LoopControl specified in the DiagnosticSetting parameter of the RunDiagnosticService
1446 method does not match any SupportedLoopControl values specified in the DiagnosticServiceCapabilities
1447 and was ignored.

1448 This alert would be sent if a DiagnosticServiceCapabilities exists for the DiagnosticTest and the client
1449 asked for a LoopControl that was not included in the SupportedLoopControl property. The LoopControl
1450 was ignored and the test ran without that control. If multiple LoopControls were specified and missing
1451 from the capabilities, there would be one alert message for each LoopControl.

1452 The variables in this message are:

- 1453 • Loop Control – Identifies the LoopControl value that was ignored.
- 1454 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1455 property of the DiagnosticTest instance.
- 1456 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1457 that was specified.

1458 This could be one of the following:

- 1459 – The Object Path of the element
- 1460 – The ElementName of the element
- 1461 – A unique user friendly name not in the model (such as, asset name)

1462 The Element Moniker can be any of these, but whichever one is used shall be used consistently
 1463 for all managed elements of the same type within the scoping profile (such as, all disk drives in
 1464 a system).

1465 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
 1466 Capabilities Mismatch.

1467 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1468 **7.9.22 DIAG28 – Capability to set LogStorage not supported**

1469 The test ran, but a LogStorage specified in the DiagnosticSetting parameter of the RunDiagnosticService
 1470 method does not match any SupportedLogStorage values specified in the DiagnosticServiceCapabilities
 1471 and was ignored.

1472 This alert would be sent if a DiagnosticServiceCapabilities exists for the DiagnosticTest and the client
 1473 asked for a LogStorage that was not included in the SupportedLogStorage property. The LogStorage was
 1474 ignored and the test ran without that log option. If multiple LogStorage values were specified and missing
 1475 from the capabilities, there would be one alert message for each LogStorage not in the
 1476 SupportedLogStorage property.

1477 The variables in this message are:

- 1478 • LogStorage Option – Identifies the type of log storage that is not supported by the capabilities.
- 1479 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
 1480 property of the DiagnosticTest instance.
- 1481 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
 1482 that was specified.

1483 This could be one of the following:

- 1484 – The Object Path of the element
- 1485 – The ElementName of the element
- 1486 – A unique user friendly name not in the model (such as, asset name)

1487 The Element Moniker can be any of these, but whichever one is used shall be used consistently
 1488 for all managed elements of the same type within the scoping profile (such as, all disk drives in
 1489 a system).

1490 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
 1491 Capabilities Mismatch.

1492 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1493 **7.9.23 DIAG30 – Capability to set PercentOfTestCoverage not supported**

1494 The test ran, but the PercentOfTestCoverage option specified in the DiagnosticSetting parameter of the
 1495 RunDiagnosticService method is not included in the SupportedServiceModes specified in the
 1496 DiagnosticServiceCapabilities and was ignored.

1497 This alert would be sent if a DiagnosticServiceCapabilities exists for the DiagnosticTest and the client
 1498 asked for a PercentOfTestCoverage, but PercentOfTestCoverage was not included in the
 1499 SupportedServiceModes property. The PercentOfTestCoverage was ignored and the test ran without that
 1500 option.

1501 The variables in this message are:

- 1502 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1503 property of the DiagnosticTest instance.
- 1504 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1505 that was specified.

1506 This could be one of the following:

- 1507 – The Object Path of the element
- 1508 – The ElementName of the element
- 1509 – A unique user friendly name not in the model (such as, asset name)

1510 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1511 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1512 a system).

1513 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
1514 Capabilities Mismatch.

1515 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1516 **7.9.24 DIAG31 – Capability to set QuickMode not supported**

1517 The test ran, but the QuickMode option specified in the DiagnosticSetting parameter of the
1518 RunDiagnosticService method is not included in the SupportedServiceModes specified in the
1519 DiagnosticServiceCapabilities and was ignored.

1520 This alert would be sent if a DiagnosticServiceCapabilities exists for the DiagnosticTest and the client
1521 asked for QuickMode, but QuickMode was not included in the SupportedServiceModes property. The
1522 QuickMode was ignored and the test ran without that option.

1523 The variables in this message are:

- 1524 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1525 property of the DiagnosticTest instance.
- 1526 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1527 that was specified.

1528 This could be one of the following:

- 1529 – The Object Path of the element
- 1530 – The ElementName of the element
- 1531 – A unique user friendly name not in the model (such as, asset name)

1532 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1533 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1534 a system).

1535 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
1536 Capabilities Mismatch.

1537 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1538 **7.9.25 DIAG32 – Capability to set HaltOnError not supported**

1539 The test ran, but the HaltOnError option specified in the DiagnosticSetting parameter of the
1540 RunDiagnosticService method is not included in the SupportedServiceModes specified in the
1541 DiagnosticServiceCapabilities and was ignored.

1542 This alert would be sent if a DiagnosticServiceCapabilities exists for the DiagnosticTest and the client
 1543 asked for the HaltOnError option, but HaltOnError was not included in the SupportedServiceModes
 1544 property. The HaltOnError was ignored and the test ran without that option.

1545 The variables in this message are:

- 1546 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
 1547 property of the DiagnosticTest instance.
- 1548 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
 1549 that was specified.

1550 This could be one of the following:

- 1551 – The Object Path of the element
- 1552 – The ElementName of the element
- 1553 – A unique user friendly name not in the model (such as, asset name)

1554 The Element Moniker can be any of these, but whichever one is used shall be used consistently
 1555 for all managed elements of the same type within the scoping profile (such as, all disk drives in
 1556 a system).

1557 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
 1558 Capabilities Mismatch.

1559 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1560 **7.9.26 DIAG33 – Capability to set NonDestructive to true not supported**

1561 The test was not run because the DiagnosticSetting NonDestructive was set to TRUE but the
 1562 DiagnosticServiceCapabilities.SupportedServiceModes does not include NonDestructive.

1563 This alert would be sent if the client supplied a DiagnosticSetting parameter to the RunDiagnosticService
 1564 with NonDestructive set to TRUE, but the DiagnosticServiceCapabilities.SupportedServiceModes does
 1565 not include NonDestructive. The test was not run because it might be destructive.

1566 The variables in this message are:

- 1567 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
 1568 property of the DiagnosticTest instance.
- 1569 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
 1570 that was specified.

1571 This could be one of the following:

- 1572 – The Object Path of the element
- 1573 – The ElementName of the element
- 1574 – A unique user friendly name not in the model (such as, asset name)

1575 The Element Moniker can be any of these, but whichever one is used shall be used consistently
 1576 for all managed elements of the same type within the scoping profile (such as, all disk drives in
 1577 a system).

1578 With this alert, the AlertType shall have the value 4 (Process Error).

1579 With this alert, the PerceivedSeverity shall have the value 5 (Major).

1580 **7.9.27 DIAG43 – The Requested DiagnosticSettings is not supported**

1581 The test ran, but the requested DiagnosticSettings property parameter of the RunDiagnosticService
 1582 method is not supported and was not used.

1583 This alert would be sent if a DiagnosticSettings property requested in the RunDiagnosticService extrinsic
1584 method is not supported for the test or the element referenced.

1585 The variables in this message are:

- 1586 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1587 property of the DiagnosticTest instance.

- 1588 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1589 that was specified.

1590 This could be one of the following:

- 1591 – The Object Path of the element
- 1592 – The ElementName of the element
- 1593 – A unique user friendly name not in the model (such as, asset name)

1594 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1595 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1596 a system).

- 1597 • DiagnosticSettings Property – Identifies the DiagnosticSettings property by property name.

- 1598 • DiagnosticSettings Value – Identifies the value requested.

- 1599 • DiagnosticSettings Used – Identifies the value used instead of the requested value.

1600 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
1601 Parameter Ignored.

1602 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1603 **7.9.28 DIAG44 – The test did not start**

1604 The test did not start for one of a variety of reasons.

1605 This alert would be sent as a test completion status message. The reason for why the test did not start
1606 would be identified by an earlier alert message (or in the log). For example, DIAG12 (see [DSP1119](#)) is an
1607 example of a message that might have been sent earlier.

1608 The variables in this message are:

- 1609 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1610 property of the DiagnosticTest instance.

- 1611 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1612 that was specified.

1613 This could be one of the following:

- 1614 – The Object Path of the element
- 1615 – The ElementName of the element
- 1616 – A unique user friendly name not in the model (such as, asset name)

1617 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1618 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1619 a system).

- 1620 • Log Object Path – This would be the Object Path of the CIM_DiagnosticLog instance.

1621 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to Test Not
1622 Started.

1623 With this alert, the PerceivedSeverity shall have the value 2 (Information).

1624 **7.9.29 DIAG45 – The test aborted**

1625 The test was not completed for various reasons.

1626 This alert would be sent as a test completion status message. The reason for why the test aborted would
 1627 be identified by an earlier alert message (or in the log). For example, “The test was killed by the client”
 1628 (see DIAG19 in [DSP1119](#)) and “The test was terminated by client” (see DIAG20 in [DSP1119](#)) are two
 1629 messages that might precede this message.

1630 The variables in this message are:

- 1631 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
 1632 property of the DiagnosticTest instance.
- 1633 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
 1634 that was specified.

1635 This could be one of the following:

- 1636 – The Object Path of the element
- 1637 – The ElementName of the element
- 1638 – A unique user friendly name not in the model (such as, asset name)

1639 The Element Moniker can be any of these, but whichever one is used shall be used consistently
 1640 for all managed elements of the same type within the scoping profile (such as, all disk drives in
 1641 a system).

- 1642 • Log Object Path – Identifies the Object Path of the CIM_DiagnosticLog instance.

1643 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to Test
 1644 Aborted.

1645 With this alert, the PerceivedSeverity shall have the value 2 (Information).

1646 **7.9.30 DIAG46 – LogStorage mismatch with capabilities**

1647 The test ran, but a LogStorage request was not one that was identified in the
 1648 DiagnosticServiceCapabilities.

1649 This alert would be sent if the client requested one or more log storage types, but one of them is not
 1650 identified in the DiagnosticServiceCapabilities.

1651 NOTE If multiple types are not supported by the capabilities, multiple alerts would be sent. This does not report
 1652 cases where the LogStorage is not supported for other reasons. That situation is handled by a separate alert (see
 1653 7.9.13).

1654 The variables in this message are:

- 1655 • Log Storage Requested – Identifies the LogStorage requested.
- 1656 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
 1657 property of the DiagnosticTest instance.
- 1658 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
 1659 that was specified.

1660 This could be one of the following:

- 1661 – The Object Path of the element
- 1662 – The ElementName of the element
- 1663 – A unique user friendly name not in the model (such as, asset name)

1664 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1665 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1666 a system).

1667 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
1668 Parameter Ignored.

1669 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1670 **7.9.31 DIAG47 – Capability to set the DiagnosticSettings parameter not supported**

1671 The test ran, but a property in the DiagnosticSettings input to the RunDiagnosticService method was not
1672 supported and was ignored.

1673 This alert would be sent if client attempted to set a DiagnosticSettings property that cannot be set.

1674 The variables in this message are:

- 1675 • Diag Setting Property – Identifies the property that was set, but not supported.
- 1676 • Diag Setting Property Value – Identifies the value supplied for the property.
- 1677 • Diagnostic Test Name – Identifies the Diagnostic Test instance that was run. This is the Name
1678 property of the DiagnosticTest instance.
- 1679 • Element Moniker – Identifies a unique name for the element under test (such as, Disk Drive)
1680 that was specified.

1681 This could be one of the following:

- 1682 – The Object Path of the element
- 1683 – The ElementName of the element
- 1684 – A unique user friendly name not in the model (such as, asset name)

1685 The Element Moniker can be any of these, but whichever one is used shall be used consistently
1686 for all managed elements of the same type within the scoping profile (such as, all disk drives in
1687 a system).

1688 With this alert, the AlertType shall have the value 1 (Other). The OtherAlertType should be set to
1689 Parameter Ignored.

1690 With this alert, the PerceivedSeverity shall have the value 3 (Warning).

1691 **8 Methods**

1692 This clause details the requirements for supporting intrinsic operations and extrinsic methods for the CIM
1693 elements defined by this profile.

1694 **8.1 CIM_DiagnosticService.RunDiagnosticService() extrinsic method**

1695 The RunDiagnosticService() method is invoked to commence execution of a diagnostic service on a
1696 specific instance of a managed element. The input parameters specify this managed element and the
1697 settings that are to be applied to the diagnostic service and the resultant job. The method returns a
1698 reference to the CIM_ConcreteJob instance that is created.

1699 Before invoking this method, clients examine the appropriate capabilities and create valid
1700 CIM_DiagnosticSettingData and CIM_JobSettingData instances to apply as input parameters. The
1701 RunDiagnosticService() method shall capture the attributes of CIM_DiagnosticSettingData in an instance
1702 of CIM_DiagnosticSettingDataRecord. This information is useful for post-mortem analysis of diagnostic
1703 results.

1704 A job shall be instantiated to run and monitor the diagnostic service. The job shall also provide useful
 1705 accounting and status information when the diagnostic service has been completed.

1706 RunDiagnosticService() return values are specified in Table 2 and parameters are specified in Table 3.
 1707 No standard messages are defined.

1708 **Table 2 – RunDiagnosticService() method: Return code values**

Value	Description
0	Job completed with no error
2	Unknown or unspecified error
3	Cannot complete within the timeout period
4	Failed
5	Invalid parameter
0x8000..0xFFFF	Vendor specific

1709 **Table 3 – RunDiagnosticService() method: Parameters**

Qualifiers	Name	Type	Description/Values
IN	ManagedElement	CIM_ManagedElement	A reference that specifies the element upon which to run the diagnostic service
IN	DiagnosticSettings	[EmbeddedInstance(CIM_DiagnosticSettingData)] string	A string (encoding a CIM_DiagnosticSettingData instance) that specifies the settings to be applied to the diagnostic service. If NULL, the diagnostic service's defaults are used.
IN	JobSettings	[EmbeddedInstance(CIM_JobSettingData)] string	A string (encoding a CIM_JobSettingData instance) that specifies the settings to be applied to the resulting job. If NULL, the job's defaults are used.
OUT	Job	CIM_ConcreteJob	A reference to the resulting job

1710 **8.2 CIM_Log.ClearLog() extrinsic method**

1711 The ClearLog() method is invoked to delete all records (instances of CIM_DiagnosticRecord subclasses)
 1712 that are associated with the log instance through the CIM_LogManagesRecord association. This method
 1713 has no parameters, and no standard messages are defined.

1714 ClearLog return values are specified in Table 4.

1715 **Table 4 – ClearLog() method: Return code values**

Value	Description
0	Request was successfully executed
2	Unknown or unspecified error
3	Cannot be completed within the timeout period
4	Failed
5	Invalid parameter
0x8000..0xFFFF	Vendor specific

1716 8.3 CIM_HelpService.GetHelp() extrinsic method

1717 The GetHelp() method is invoked to obtain documentation about a diagnostic service. The input
1718 parameters provide the name, format, and delivery type of a document.

1719 The CIM_HelpService class has some attributes that publish the available documents, supported delivery
1720 types, and formats. See Table 6 for additional information. Before invoking this method, clients check
1721 these attributes in order to request an available document, format, and delivery type.

1722 GetHelp() return values are specified in Table 5 and parameters are specified in Table 6. No standard
1723 messages are defined.

1724 **Table 5 – GetHelp() method: Return code values**

Value	Description
0	Request was successfully executed
2	Unknown or unspecified error
3	Cannot be completed within the timeout period
4	Failed
5	Invalid parameter
0x1000	Busy — indicates that the method cannot be invoked "at this time" It is not an error condition, but signals that the implementation is doing something else and cannot respond.
0x1001	Requested document not found
0x8000..0xFFFF	Vendor Reserved

1725 **Table 6 – GetHelp() method: Parameters**

Qualifiers	Name	Type	Description/Values
IN	RequestedDocument	string	The document that should be made available to the client. The available documents are published in the DocumentsAvailable attribute.
IN	Format	uint16	The format that the document should have. The supported formats are published in the DocumentFormat attribute.
IN	RequestedDelivery	uint16	The way in which the document should be made available (fully specified path, launch a program, file contents, and so on).
OUT	DocumentInfo	string	This parameter returns information about the document. The format and content will depend on the RequestedDelivery parameter.

1726 8.4 Profile conventions for operations

1727 For each profile class (including associations), the implementation requirements for operations, including
1728 those in the following default list, are specified in class-specific subclauses of this clause.

1729 The default list of operations is as follows:

- 1730 • GetInstance
- 1731 • EnumerateInstances
- 1732 • EnumerateInstanceNames
- 1733 • Associators
- 1734 • AssociatorNames
- 1735 • References
- 1736 • ReferenceNames

1737 **8.5 CIM_DiagnosticTest**

1738 Table 7 lists implementation requirements for operations. If implemented, these operations shall be
 1739 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 7, all operations in
 1740 the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1741 NOTE Related profiles may define additional requirements on operations for the profile class.

1742 **Table 7 – Operations: CIM_DiagnosticTest**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
InvokeMethod	Mandatory	None
ExecQuery	Optional	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None

1743 **8.6 CIM_AvailableDiagnosticService**

1744 Table 8 lists implementation requirements for operations. If implemented, these operations shall be
 1745 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 8, all operations in
 1746 the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1747 NOTE Related profiles may define additional requirements on operations for the profile class.

1748 **Table 8 – Operations: CIM_AvailableDiagnosticService**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1749 8.7 CIM_ServiceAffectsElement

1750 Table 9 lists implementation requirements for operations. If implemented, these operations shall be
 1751 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 9, all operations in
 1752 the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1753 NOTE Related profiles may define additional requirements on operations for the profile class.

1754 **Table 9 – Operations: CIM_ServiceAffectsElement**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1755 8.8 CIM_SoftwareIdentity

1756 Table 10 lists implementation requirements for operations. If implemented, these operations shall be
 1757 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 10, all operations
 1758 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1759 NOTE Related profiles may define additional requirements on operations for the profile class.

1760 **Table 10 – Operations: CIM_SoftwareIdentity**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
ExecQuery	Optional	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None

1761 8.9 CIM_ElementSoftwareIdentity

1762 Table 11 lists implementation requirements for operations. If implemented, these operations shall be
 1763 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 11, all operations
 1764 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1765 NOTE Related profiles may define additional requirements on operations for the profile class.

1766 **Table 11 – Operations: CIM_ElementSoftwareIdentity**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1767 **8.10 CIM_HelpService**

1768 Table 12 lists implementation requirements for operations. If implemented, these operations shall be
 1769 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 12, all operations
 1770 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1771 NOTE Related profiles may define additional requirements on operations for the profile class.

1772 **Table 12 – Operations: CIM_HelpService**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
InvokeMethod	Mandatory	None
ExecQuery	Optional	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None

1773 **8.11 CIM_ServiceAvailableToElement**

1774 Table 13 lists implementation requirements for operations. If implemented, these operations shall be
 1775 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 13, all operations
 1776 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1777 NOTE Related profiles may define additional requirements on operations for the profile class.

1778 **Table 13 – Operations: CIM_ServiceAvailableToElement**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1779 **8.12 CIM_DiagnosticSettingData**

1780 Table 14 lists implementation requirements for operations. If implemented, these operations shall be
 1781 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 14, all operations
 1782 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1783 NOTE Related profiles may define additional requirements on operations for the profile class.

1784

Table 14 – Operations: CIM_DiagnosticSettingData

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
ExecQuery	Optional	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None

1785 8.13 CIM_DiagnosticServiceCapabilities

1786 Table 15 lists implementation requirements for operations. If implemented, these operations shall be
 1787 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 15, all operations
 1788 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1789 NOTE Related profiles may define additional requirements on operations for the profile class.

1790

Table 15 – Operations: CIM_DiagnosticServiceCapabilities

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
ExecQuery	Optional	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None

1791 8.14 CIM_ElementCapabilities

1792 Table 16 lists implementation requirements for operations. If implemented, these operations shall be
 1793 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 16, all operations
 1794 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1795 NOTE Related profiles may define additional requirements on operations for the profile class.

1796

Table 16 – Operations: CIM_ElementCapabilities

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1797 **8.15 CIM_ElementSettingData**

1798 Table 17 lists implementation requirements for operations. If implemented, these operations shall be
 1799 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 17, all operations
 1800 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1801 NOTE Related profiles may define additional requirements on operations for the profile class.

1802 **Table 17 – Operations: CIM_ElementSettingData**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1803 **8.16 CIM_DiagnosticLog**

1804 Table 18 lists implementation requirements for operations. If implemented, these operations shall be
 1805 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 18, all operations
 1806 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1807 NOTE Related profiles may define additional requirements on operations for the profile class.

1808 **Table 18 – Operations: CIM_DiagnosticLog**

Operation	Requirement	Messages
DeleteInstance	Optional	None
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
InvokeMethod	Mandatory	None
ExecQuery	Optional	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None

1809 **8.16.1 DeleteInstance**

1810 DeleteInstance shall be supported if the implementation supports CIM_DiagnosticLog and allows the
 1811 CIM_DiagnosticSettingData.ResultPersistence to be set to 0xFFFFFFFF (“Persist Forever”). This allows
 1812 the client to delete the log and all its records with one operation on the log.

1813 **8.17 CIM_UseOfLog**

1814 Table 19 lists implementation requirements for operations. If implemented, these operations shall be
 1815 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 19, all operations
 1816 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1817 NOTE Related profiles may define additional requirements on operations for the profile class.

1818

Table 19 – Operations: CIM_UseOfLog

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1819 8.18 CIM_DiagnosticServiceRecord

1820 Table 20 lists implementation requirements for operations. If implemented, these operations shall be
 1821 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 20, all operations
 1822 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1823 NOTE Related profiles may define additional requirements on operations for the profile class.

1824

Table 20 – Operations: CIM_DiagnosticServiceRecord

Operation	Requirement	Messages
GetInstance	Mandatory	None
DeleteInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
ExecQuery	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None

1825 8.18.1 DeleteInstance

1826 DeleteInstance shall be supported if the implementation supports DiagnosticServiceRecord and wants to
 1827 give the client the ability to delete records after it has read them and stored them in client storage. This
 1828 may be required if the test generates a lot of records and the test is at risk of running out of resources.

1829 8.19 CIM_DiagnosticCompletionRecord

1830 Table 21 lists implementation requirements for operations. If implemented, these operations shall be
 1831 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 21, all operations
 1832 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1833 NOTE Related profiles may define additional requirements on operations for the profile class.

1834

Table 21 – Operations: CIM_DiagnosticCompletionRecord

Operation	Requirement	Messages
GetInstance	Mandatory	None
DeleteInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
ExecQuery	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None

1835 **8.19.1 DeleteInstance**

1836 DeleteInstance shall be supported if the implementation supports DiagnosticCompletionRecord and
 1837 wants to give the client the ability to delete records after it has read them and stored them in client
 1838 storage. This may be required if the test generates a lot of records and the test is at risk of running out of
 1839 resources.

1840 **8.20 CIM_DiagnosticSettingDataRecord**

1841 Table 22 lists implementation requirements for operations. If implemented, these operations shall be
 1842 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 22, all operations
 1843 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1844 NOTE Related profiles may define additional requirements on operations for the profile class.

1845

Table 22 – Operations: CIM_DiagnosticSettingDataRecord

Operation	Requirement	Messages
GetInstance	Mandatory	None
DeleteInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
ExecQuery	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Optional	None
ReferenceNames	Optional	None

1846 **8.20.1 DeleteInstance**

1847 DeleteInstance shall be supported if the implementation supports DiagnosticSettingDataRecord and
 1848 wants to give the client the ability to delete records after it has read them and stored them in client
 1849 storage. This may be required if the test generates a lot of records and the test is at risk of running out of
 1850 resources.

1851 8.21 CIM_LogManagesRecord

1852 Table 23 lists implementation requirements for operations. If implemented, these operations shall be
 1853 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 23, all operations
 1854 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1855 NOTE Related profiles may define additional requirements on operations for the profile class.

1856 **Table 23 – Operations: CIM_LogManagesRecord**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1857 8.22 CIM_RecordAppliesToElement

1858 Table 24 lists implementation requirements for operations. If implemented, these operations shall be
 1859 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 24, all operations
 1860 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1861 NOTE Related profiles may define additional requirements on operations for the profile class.

1862 **Table 24 – Operations: CIM_RecordAppliesToElement**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1863 8.23 CIM_CorrespondingSettingDataRecord

1864 Table 25 lists implementation requirements for operations. If implemented, these operations shall be
 1865 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 25, all operations
 1866 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1867 NOTE Related profiles may define additional requirements on operations for the profile class.

1868 **Table 25 – Operations: CIM_CorrespondingSettingDataRecord**

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1869 8.24 CIM_ServiceComponent

1870 Table 26 lists implementation requirements for operations. If implemented, these operations shall be
 1871 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 26, all operations
 1872 in the default list in 8.4 shall be implemented as defined in [DSP0200](#).

1873 NOTE Related profiles may define additional requirements on operations for the profile class.

1874

Table 26 – Operations: CIM_ServiceComponent

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

1875

1876 **9 Use cases**

1877 This clause contains object diagrams and use cases for the *Diagnostics Profile*.

1878 **9.1 Profile conformance**

1879 Conformance of a central class instance and its associated instances to a particular profile may be
1880 identified by examining instances of the CIM_ElementConformsToProfile association class according to
1881 the Central Class Methodology. In some environments, an alternative method that relies on the Scoping
1882 Class Methodology through the scoping class instance may be desirable.

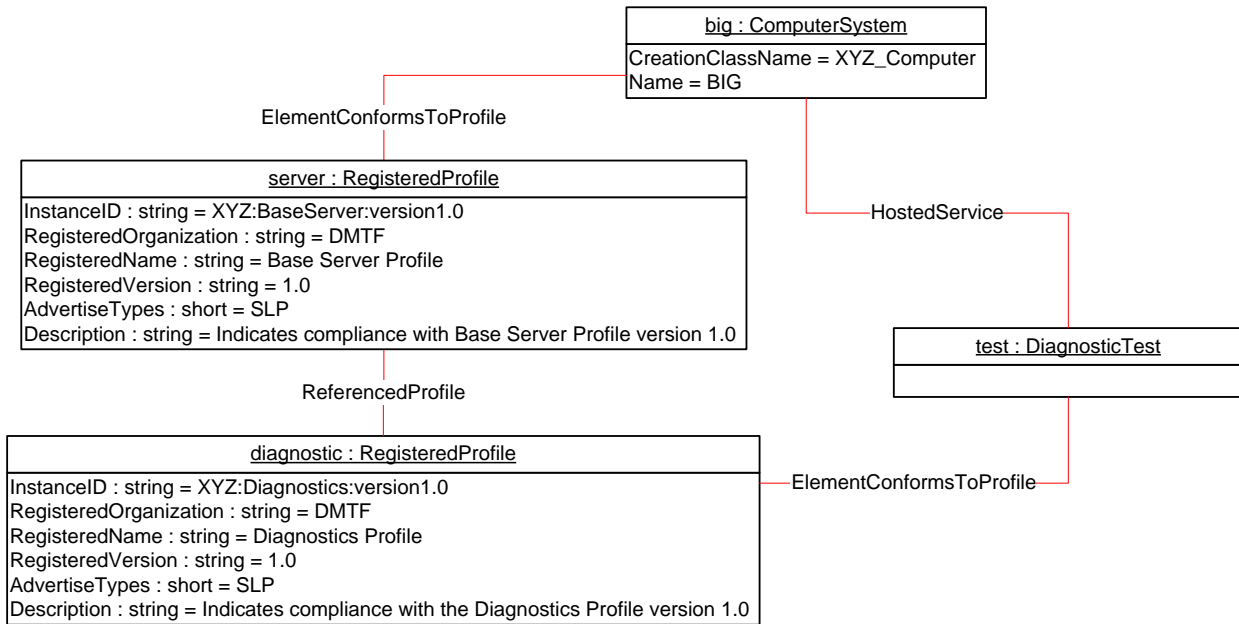
1883 With CIM_ComputerSystem as the Scoping Class of this profile, the object diagram in Figure 2 shows
1884 how instances of CIM_RegisteredProfile may be used to identify the version of the *Diagnostics Profile* to
1885 which an instance of CIM_DiagnosticTest and its associated instances conform. In this example (using
1886 BaseServer as the system configuration), one instance of CIM_RegisteredProfile identifies the "*Base
1887 Server Profile v1.0*" and the other instance identifies the "*Diagnostics Profile v2.0*."

1888 To support the Scoping Class Methodology for advertising profile implementation conformance, a
1889 CIM_DiagnosticTest instance is associated to an instance of the Scoping Class, CIM_ComputerSystem,
1890 through an instance of CIM_HostedService. This instance of CIM_ComputerSystem is advertised as
1891 being in implementation conformance with the *Base Server Profile v1.0* as indicated by the
1892 CIM_ElementConformsToProfile association to the "server" CIM_RegisteredProfile instance. The
1893 CIM_ReferencedProfile relationship between "server" and "diagnostic" places the CIM_DiagnosticTest
1894 instance within the scope of "diagnostic." Thus, the CIM_DiagnosticTest instance is conformant with the
1895 *Diagnostics Profile v2.0*.

1896 To support the Central Class Methodology for advertising profile implementation conformance, a
1897 CIM_ElementConformsToProfile association is established between the CIM_DiagnosticTest central
1898 class instance and the instance of CIM_RegisteredProfile that represents the *Diagnostics Profile*.

1899 For these methodologies to be successful, profiles for systems that can support diagnostics need to
1900 reference the *Diagnostics Profile*. In this example, the [Base Server Profile](#) would need to include the
1901 *Diagnostics Profile* in its "Related profiles" table.
1902

1903 The CIM_ prefix has been omitted from the class names in Figure 2 for simplicity and readability.



1904

1905

Figure 2 – Registered profile

1906 **9.2 Use case summary**

1907 Table 27 summarizes the use cases that are described in this clause. The use cases are categorized and
 1908 named, and references are provided to the body text that describes the use case.

1909 NOTE Although use case names follow the convention for naming classes, properties, and methods in the schema,
 1910 this naming was done for readability only and does not imply any functionality attached to the name.

1911 The CIM_ prefix has been omitted from the class names in the use cases for readability.

1912

Table 27 – Diagnostics Profile use cases

Category	Name	Description
Discover Available Diagnostics See 9.4.	GetAllDiagnostics	Find all diagnostics available on a system. See 9.4.1.
	GetAllDiagnosticMEPairs	Find all diagnostic/managed elements pairs available on a system. See 9.4.2.
	GetDiagnosticsForME	Find all the diagnostics available on a system for a managed element. See 9.4.3.
	GetMEsForDiagnostic	Find all the managed elements that support a particular diagnostic. See 9.4.4.
	GetCapabilitiesOfDiagnostic	Find the capabilities of a particular diagnostic. See 9.4.5.
	GetCharacteristicsOfDiagnostic	Find the characteristics of a particular diagnostic. See 9.4.6.

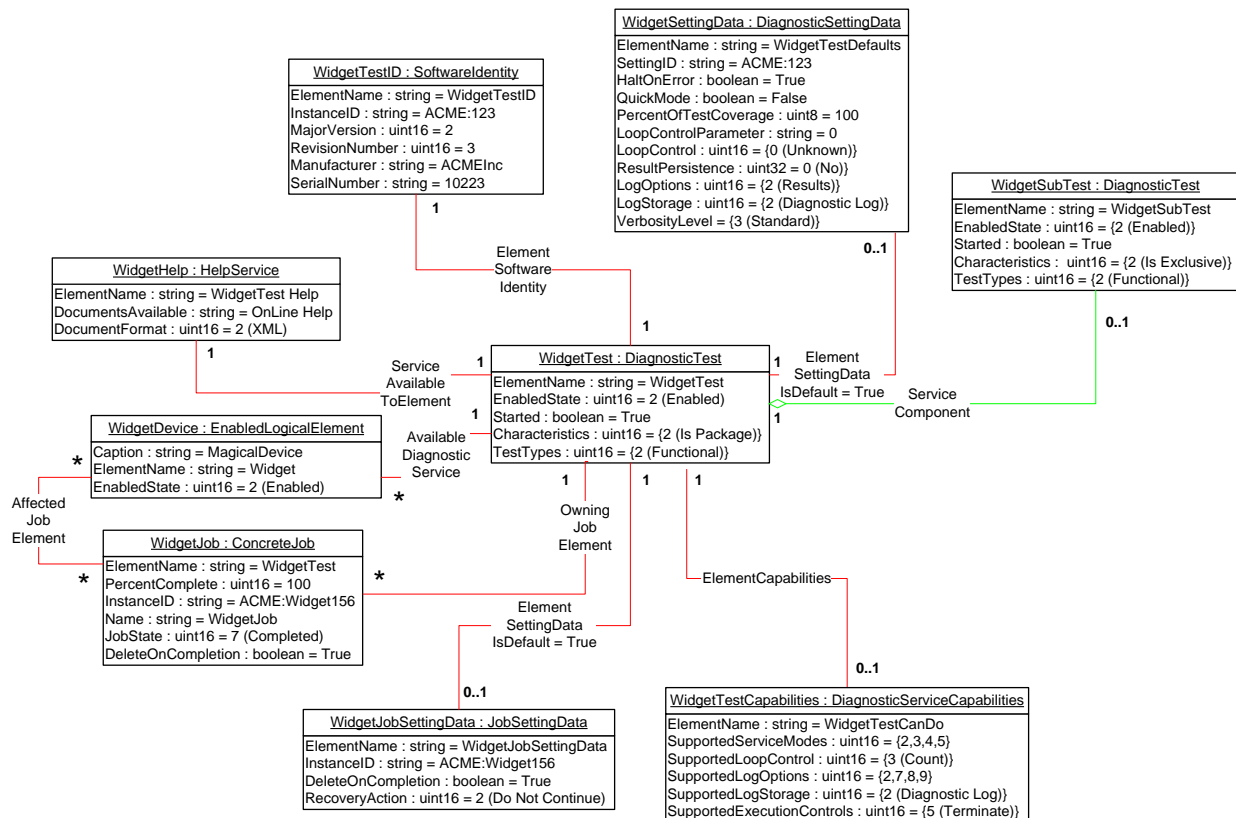
Category	Name	Description
	GetDiagnosticsWithCharacteristicsForME	Find all the diagnostics available on a system, for a managed element, with certain characteristics. See 9.4.7.
	GetDiagnosticsWithCapabilitiesForME	Find all the diagnostics available on a system, for a managed element, with certain capabilities. See 9.4.8.
	GetPackageSubtests	Find the subtests for a diagnostic test with the value of the DiagnosticTest.Characteristics property set to Is Package. See 9.4.9.
Configure Diagnostic See 9.5.	GetDefaultDiagnosticSettings	Find the default diagnostic settings for a diagnostic. See 9.5.1.
	CreateDiagnosticSettings	Create a unique setting for a diagnostic. See 9.5.2.
	GetDefaultJobSettings	Find the default job settings for a diagnostic. See 9.5.3.
	CreateJobSettings	Create a unique setting for a diagnostic job. See 9.5.4.
Execute and Control Diagnostic See 9.6.	RunDiagnostic	Run a diagnostic with default and unique settings. See 9.6.1.
	SuspendDiagnostic	Suspend a running diagnostic. See 9.6.2.
	ResumeDiagnostic	Resume a suspended diagnostic. See 9.6.3.
	AbortDiagnostic	Abort a running diagnostic. See 9.6.4.
	KillDiagnostic	Abort a running diagnostic immediately, with no attempt to perform a clean shutdown. See 9.6.5.
Discover Diagnostic Executions See 9.7.	GetAffectedMEs	Find all the managed elements affected by a running diagnostic. See 9.7.1.
	GetAllDiagnosticExecutionsForME	Find all the diagnostic executions on a system for a managed element. See 9.7.2.
	GetSpecificDiagnosticExecutions	Find all the executions of a specific diagnostic. See 9.7.3.
	GetSpecificDiagnosticExecutionsForME	Find all the executions of a specific diagnostic for a particular managed element. See 9.7.4.
Discover Diagnostic Results (in-progress and final) See 9.8.	GetLogRecordsForDiagnostic	Find all the diagnostic log records for a particular diagnostic. See 9.8.1.
	GetLogRecordsForME	Find all the diagnostic log records for a particular managed element. See 9.8.2.

Category	Name	Description
	GetLogRecordsForMEAndDiagnostic	Find all the diagnostic log records for a particular diagnostic run on a particular managed element. See 9.8.3.
	GetDiagnosticExecutionFinalResults	Determine the final result of a diagnostic execution. See 9.8.4.
	GetDiagnosticExecutionResults	Find all diagnostic log records for a particular execution (job). See 9.8.5.
	GetDiagnosticExecutionSettings	Find the settings used in a diagnostic execution. See 9.8.6.
	GetDiagnosticProgress	Get the progress of a running diagnostic. See 9.8.7.

1913 **9.3 Diagnostic services object diagram**

1914 Figure 3 is an object diagram for diagnostic services for a fictitious device called "Widget." Only classes, properties, and methods that are of particular interest for the diagnostic model are shown. Refer to this
 1915 diagram for the use cases in this clause.
 1916

1917 The CIM_ prefix has been omitted from the class names in the diagram for readability.



1918

1919

Figure 3 – Diagnostic services object diagram

1920 **9.4 Discover available diagnostics**

1921 The use cases in this clause describe how the client can find available diagnostics. The CIM_ prefix has
1922 been omitted from the class names in the use cases for readability.

1923 **9.4.1 GetAllDiagnostics**

1924 The client can find all of the diagnostics that are available on a system as follows:

1925 The client calls the EnumerateInstances (or EnumerateInstanceNames) operation by using the
1926 DiagnosticTest class. The operation returns DiagnosticTest instances that represent a diagnostic that is
1927 available on the system.

1928 **9.4.2 GetAllDiagnosticMEPairs**

1929 The client can find all of the diagnostics/managed element pairs that are available on a system as follows.
1930 Each pair comprises a diagnostic and a ManagedElement (device) that is supported by the diagnostic.

1931 The client calls the EnumerateInstances (or EnumerateInstanceNames) operation by using the
1932 AvailableDiagnosticService class. The operation returns AvailableDiagnosticService instances that have
1933 a reference to the DiagnosticTest instance and another reference to the ManagedElement instance.

1934 **9.4.3 GetDiagnosticsForME**

1935 The client can find all of the diagnostics on a system that can be launched against a specific device
1936 (managed element) as follows. Assume that the client starts at a known ManagedElement instance,
1937 which represents the device to be tested.

1938 From the ManagedElement instance, the client calls the Associators operation by
1939 using AvailableDiagnosticService as the association class. The operation returns DiagnosticTest
1940 instances that represent a diagnostic that can be launched against the ManagedElement.

1941 **9.4.4 GetMEsForDiagnostic**

1942 The client can find all managed elements (devices) that are supported by a specific diagnostic as follows.

1943 Assume that the client starts at a known DiagnosticTest instance. From the DiagnosticTest instance, the
1944 client calls the Associators operation by using AvailableDiagnosticService as the association class. The
1945 operation returns ManagedElement instances that represent a device that is supported by the
1946 DiagnosticTest.

1947 **9.4.5 GetCapabilitiesOfDiagnostic**

1948 A diagnostic service publishes its support for various options through a DiagnosticServiceCapabilities
1949 instance. A client can use the information in DiagnosticServiceCapabilities to generate an instance of
1950 DiagnosticSettingData that is passed as the DiagnosticSettings argument of the RunDiagnosticService
1951 extrinsic method of DiagnosticTest. The client can find the capabilities of a diagnostic as follows. Assume
1952 that the client starts at a known DiagnosticTest instance.

1953 From the DiagnosticTest instance, the client calls the Associators operation by using ElementCapabilities
1954 as the association class and DiagnosticServiceCapabilities as the result class. The operation should
1955 return only one DiagnosticServiceCapabilities instance, which represents the diagnostic capabilities.

1956 NOTE Because the implementation of DiagnosticServiceCapabilities is optional, it may not be available. In this
1957 case, no assumptions should be made regarding the diagnostic capabilities.

1958 **9.4.6 GetCharacteristicsOfDiagnostic**

1959 The client can discover all of the characteristics (is destructive, is interactive, is synchronous, and so on)
1960 of a diagnostic. From the DiagnosticTest instance, the client reads just the Characteristics and

1961 OtherCharacteristicsDescriptions attributes, which contain the diagnostic characteristics. See the MOF file
1962 class definition for DiagnosticTest for further information.

1963 **9.4.7 GetDiagnosticsWithCharacteristicsForME**

1964 The client can find all of the diagnostics that can be launched against a specific device (managed
1965 element) and have specific characteristics as follows. Assume that the client starts at a known
1966 ManagedElement instance, which represents the device to be tested.

1967 1) The client discovers all of the diagnostics that are available for the specific ManagedElement.
1968 The GetDiagnosticsForME use case (see 9.4.3) describes the necessary steps.

1969 2) For each DiagnosticTest instance, the client checks the diagnostic characteristics. The
1970 GetCharacteristicsOfDiagnostic use case (see 9.4.6) describes the necessary steps.

1971 If the characteristics of the DiagnosticTest instance match the desired characteristics, the
1972 DiagnosticTest instance is the one desired.

1973 **9.4.8 GetDiagnosticsWithCapabilitiesForME**

1974 The client can find all of the diagnostics that can be launched against a specific device (managed
1975 element) and have specific capabilities as follows. Assume that the client starts at a known
1976 ManagedElement instance, which represents the device to be tested.

1977 1) The client discovers all of the diagnostics that are available for the specific ManagedElement.
1978 The GetDiagnosticsForME use case (see 9.4.3) describes the necessary steps.

1979 2) For each DiagnosticTest instance, the client checks the diagnostic capabilities. The
1980 GetCapabilitiesOfDiagnostic use case (see 9.4.5) describes the necessary steps.

1981 If the capabilities of the DiagnosticTest instance match the desired capabilities, the
1982 DiagnosticTest instance is the one desired.

1983 **9.4.9 GetPackageSubtests**

1984 The client can find the subtests for a diagnostic test with the IsPackage value set in the
1985 DiagnosticTest.Characteristics property by using the following procedure. Assume that the client starts at
1986 a known DiagnosticTest instance.

1987 1) The client checks the DiagnosticTest.Characteristics property for the IsPackage value.

1988 2) If the IsPackage value is present, the client calls the Associators operation by using
1989 ServiceComponent as the association class and DiagnosticTest as the result class.

1990 The operation returns the DiagnosticTest instances that are subtests of the known
1991 DiagnosticTest.

1992 **9.5 Configure diagnostic**

1993 The use cases in this clause describe how the client can find and create settings for diagnostics. The
1994 CIM_ prefix has been omitted from the class names in the use cases for readability.

1995 **9.5.1 GetDefaultDiagnosticSettings**

1996 The client can obtain the default settings for a diagnostic service as follows. Assume that the client starts
1997 at a known DiagnosticTest instance.

1998 1) From the DiagnosticTest instance, the client calls the Associators operation by
1999 using ElementSettingData as the association class and DiagnosticSettingData as the result
2000 class. The operation returns DiagnosticSettingData instances.

2001 2) For each DiagnosticSettingData instance, the client calls the References operation by using
2002 ElementSettingData as the result class.

2003 The operation returns ElementSettingData instances.

2004 3) For each ElementSettingData instance, the client determines whether the value of the
2005 ElementSettingData.ManagedElement property matches the DiagnosticTest name and the
2006 value of the ElementSettingData.IsDefault property is 1 (Is Default). If so, the
2007 DiagnosticSettingData instance represents the default diagnostic settings. The name of this
2008 DiagnosticSettingData instance may also be retrieved from ElementSettingData.SettingData
2009 property.

2010 NOTE Because the implementation of DiagnosticSettingData is optional, it may not be available.

2011 9.5.2 CreateDiagnosticSettings

2012 To run a diagnostic test, the client invokes the RunDiagnosticService extrinsic method of DiagnosticTest.
2013 The DiagnosticSettings argument may be an empty string, NULL, or a string representing an embedded
2014 instance of DiagnosticSettingData. When DiagnosticSettings is an empty string or NULL, the test runs
2015 using the default settings which may or may not have been published by the implementation.

2016 Note that the diagnostic default settings are represented by a DiagnosticSettingData subclass that may
2017 have extensions. If the client is aware of the extensions, they may be modified as well. If the client is
2018 unaware, the default values should be used. Assume that the client starts at a known DiagnosticTest
2019 instance. The client may use their own diagnostic settings as follows

2020 1) The client discovers the diagnostic capabilities of the DiagnosticTest instance. The
2021 GetCapabilitiesOfDiagnostic use case (9.4.5) describes the necessary steps.

2022 2) If Step 1 does not return an instance, the client can attempt to discover the default diagnostic
2023 settings of the DiagnosticTest instance. The GetDefaultDiagnosticSettings use case (9.5.1)
2024 describes the necessary steps.

2025 3) If Step 2 does not return an instance or if the client chooses to create an instance of the
2026 DiagnosticSettingData class, a GetClass operation for DiagnosticSettingData can be performed
2027 and then used to create an instance locally in the client scope (for example, IwbemClassObject
2028 or CIMInstance object) based on the class definition.

2029 4) The client modifies the created DiagnosticSettingData instance as necessary. However, the
2030 client should consider the diagnostic capabilities during the changes. If test capabilities are
2031 published, the client should set the values in DiagnosticSettingData instance based on the
2032 published capabilities (if any) because any setting for an unsupported capability shall be
2033 ignored.

2034 9.5.3 GetDefaultJobSettings

2035 The client can obtain the default job settings for a diagnostic service as follows. Assume that the client
2036 starts at a known DiagnosticTest instance.

2037 1) From the DiagnosticTest instance, the client calls the Associators operation by
2038 using ElementSettingData as the association class and JobSettingData as the result class.

2039 2) For each JobSettingData instance that is returned, the client calls the References operation by
2040 using ElementSettingData as the result class.

2041 3) For each ElementSettingData instance that is returned, the client determines whether the value
2042 of the ElementSettingData.ManagedElement property matches the DiagnosticTest name and
2043 the value of the ElementSettingData.IsDefault property is 1 ("Is Default"). If so, the
2044 JobSettingData instance represents the default job settings. The name of this JobSettingData
2045 instance may also be retrieved from ElementSettingData.SettingData property.

2046 NOTE Because the implementation of JobSettingData is optional, it may not be available.

2047 **9.5.4 CreateJobSettings**

2048 To run a diagnostic test, the client invokes the RunDiagnosticService extrinsic method of DiagnosticTest.
 2049 The JobSettings argument may be an empty string, NULL, or a string representing an embedded instance
 2050 of JobSettingData. When JobSettings is an empty string or NULL, the job runs using the default settings
 2051 which may or may not have been published by the implementation.

2052 Note that the diagnostic default job settings are represented by a JobSettingData subclass that may have
 2053 extensions. If the client is aware of the extensions, they may be modified as well. If the client is unaware,
 2054 the default values should be used. Assume that the client starts at a known DiagnosticTest instance. The
 2055 client may use their own job settings as follows:

- 2056 1) The client can attempt to discover the default job settings of the DiagnosticTest instance. The
 2057 GetDefaultJobSettings use case (see 9.5.3) describes the necessary steps.
- 2058 2) If Step 1 does not return an instance or if the client chooses to create an instance of the
 2059 JobSettingData class, a GetClass operation for JobSettingData can be performed and then
 2060 used to create an instance locally in the client scope (for example, IwbemClassObject or
 2061 CIMInstance object) based on the class definition.
- 2062 3) The client modifies the created JobSettingData instance as necessary.

2063 **9.6 Execute and control diagnostic**

2064 The RunDiagnosticService() method is invoked to start the diagnostic service. Input parameters are the
 2065 ManagedElement being tested, the test settings, and the job settings to be used for the test execution.
 2066 The test settings and job settings arguments are optional. If the settings argument is NULL or an empty
 2067 string, the default settings are used. A reference to a ConcreteJob instance shall be returned.

2068 An instance of ConcreteJob is created by the diagnostic implementation to allow monitoring and control of
 2069 the running service. By invoking the RequestStateChange method, the client may start, stop, suspend,
 2070 and resume the job. By inspecting the value of PercentComplete, the client may determine the job's
 2071 progress.

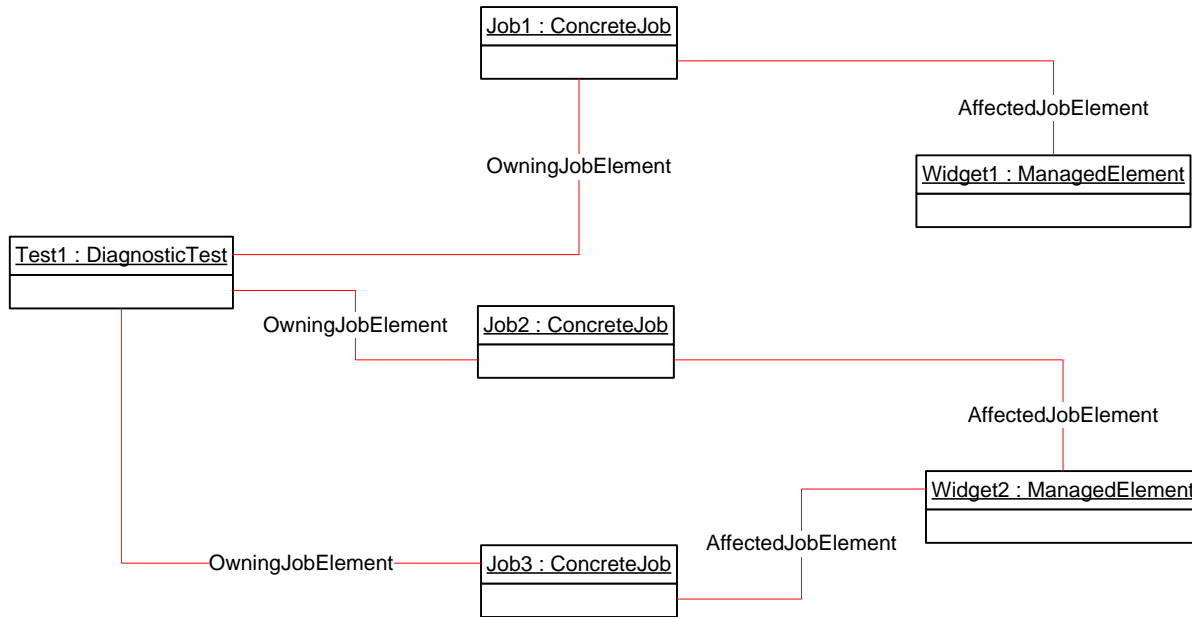
2072 The ManagedElement being tested and the DiagnosticTest instance that launched the test are related to
 2073 the job instance through the OwningJobElement and the AffectedJobElement associations. The client
 2074 may find jobs associated with services or managed elements of interest by using these associations.

2075 NOTE To expedite test data retrieval, the InstanceID values of ConcreteJob, DiagnosticSettingDataRecord,
 2076 DiagnosticServiceRecord, and DiagnosticCompletionRecord are closely related to each other. For further information,
 2077 see the Discover Diagnostic Results use cases in 9.8.

2078 Figure 4 is an object diagram that shows the state of instances when a DiagnosticTest
 2079 RunDiagnosticService() method has been called three times. Two of the times were to run a test on the
 2080 same device, ManagedElement2.

2081 NOTE Not all diagnostic tests are capable of running on the same device simultaneously; that is,
 2082 DiagnosticTest.Characteristics has the value of 2 (Is Exclusive). If this had been the case in this example, the
 2083 DiagnosticTest would have returned an error on the second RunDiagnosticService() method call to run a test on
 2084 ManagedElement2.

2085 The CIM_ prefix has been omitted from the class names in the diagram and the use cases for readability.



2086
2087 **Figure 4 – Job example**

2088 **9.6.1 RunDiagnostic**

2089 The client can run a diagnostic with default and unique settings as follows. (See 9.4 for use cases related
2090 to finding diagnostics that can be initiated. See 9.5 for use cases related to creating and modifying
2091 diagnostic settings to configure diagnostic execution.)

- 2092 1) The client calls the RunDiagnosticService() method, passing in EmbeddedInstances of
2093 DiagnosticSettingData and JobSettings to use to execute the test as well as the reference to the
2094 ManagedElement to test. If the client passes in a NULL or an empty string for these classes, the
2095 default values are used.

2096 The diagnostic service creates a Job instance to represent that test running on that managed
2097 element and shall return a reference to it in the return call from RunDiagnosticService(). In
2098 addition, the diagnostic service creates the OwningJobElement association between the Job
2099 and the Service and the AffectedJobElement association between the Job and the
2100 ManagedElement.

2101 **9.6.2 SuspendDiagnostic**

2102 The client can suspend the execution of the test by using the RequestStateChange() method call on the
2103 Job instance that is returned from the RunDiagnosticService() method, as shown in the following
2104 procedure. Assume that the client starts at a known ConcreteJob instance.

- 2105 1) The client follows the OwningJobElement association from the ConcreteJob to the
2106 DiagnosticTest
- 2107 2) The client follows the ElementCapabilities association from the DiagnosticTest to the
2108 DiagnosticServiceJobCapabilities for the service.

2109 The DiagnosticServiceJobCapabilities.RequestedStatesSupported property indicates the
2110 permitted values of the RequestedState input parameter for the
2111 ConcreteJob.RequestStateChange() extrinsic method. Because
2112 DiagnosticServiceJobCapabilities is an optional class, a client may not be able to examine an
2113 instance to determine which values of RequestedState to use. If a client invokes

- 2114 ConcreteJob.RequestStateChange() to change to an unsupported state, the extrinsic method
2115 shall return 4097 (Invalid State Transition).
- 2116 3) The client checks the DiagnosticServiceJobCapabilities.RequestedStatesSupported property for
2117 the value of 3 (Suspend).
- 2118 If the value exists, the Job supports suspension.
- 2119 4) The client should cache the capabilities for the DiagnosticTest for future reference.
- 2120 5) Assuming the job supports the suspend operation, the client calls the RequestStateChange()
2121 method for the ConcreteJob instance, passing in a RequestedState value of 3 (Suspend).
- 2122 After the transition is completed successfully, the ConcreteJob that represents the test will set
2123 the value of the JobState property to 5 (Suspended) and set the value of
2124 TimeOfLastStateChange to the current time.

2125 **9.6.3 ResumeDiagnostic**

2126 The client can resume the execution of a test by using the RequestStateChange() method call on the Job
2127 instance that is returned from the RunDiagnosticService() method, as shown in the following procedure.
2128 Assume that the client starts at a known DiagnosticTest instance.

- 2129 1) The client follows the ElementCapabilities association from the DiagnosticTest to the
2130 DiagnosticServiceJobCapabilities for the service.
- 2131 The DiagnosticServiceJobCapabilities.RequestedStatesSupported property indicates the
2132 permitted values of the RequestedState input parameter for the
2133 ConcreteJob.RequestStateChange() extrinsic method. Because
2134 DiagnosticServiceJobCapabilities is an optional class, a client may not be able to examine an
2135 instance to determine which values of RequestedState to use. If a client invokes
2136 ConcreteJob.RequestStateChange() to change to an unsupported state, the extrinsic method
2137 shall return 4097 (Invalid State Transition).
- 2138 2) The client checks the DiagnosticServiceJobCapabilities.RequestedStatesSupported property for
2139 the value of 2 (Start).
- 2140 If the value exists, the Job supports resumption.
- 2141 3) The client finds the appropriate Job instances. The GetSpecificDiagnosticExecutions use case
2142 (see 9.7.3) describes the necessary steps.
- 2143 4) The client calls the RequestStateChange() method of DiagnosticTest, passing in a
2144 RequestedState value of 2 (Start).
- 2145 After the transition is completed successfully, the ConcreteJob that represents the test will set
2146 the value of the JobState property to 4 (Running) and set the value of TimeOfLastStateChange
2147 to the current time.

2148 NOTE The JobState property may transition from the value 3 (Starting) before the final transition to the value of 4
2149 (Running).

2150 **9.6.4 AbortDiagnostic**

2151 The client can cleanly abort the execution of a test by using the RequestStateChange() method call on
2152 the Job instance that is returned from the RunDiagnosticService() method, as shown in the following
2153 procedure. Assume that the client starts at a known DiagnosticTest instance.

- 2154 1) The client follows the ElementCapabilities association from the DiagnosticTest to the
2155 DiagnosticServiceJobCapabilities for the service.
- 2156 If no DiagnosticServiceJobCapabilities is returned, proceed to step 3. Support for Terminate is
2157 mandatory.

2158 2) The client checks the DiagnosticServiceJobCapabilities.RequestedStatesSupported property for
2159 the value of 4 (Terminate).

2160 If the value exists, the Job supports termination.

2161 3) The client finds the appropriate Job instances. The GetSpecificDiagnosticExecutions use case
2162 (see 9.7.3) describes the necessary steps.

2163 4) The client calls the RequestStateChange() method, passing in a RequestedState value of 4
2164 (Terminate).

2165 After the transition is completed successfully, the ConcreteJob that represents the test will set
2166 the value of the JobState property to 8 (Terminated) and set the value of
2167 TimeOfLastStateChange to the current time.

2168 NOTE The JobState property may transition to **Number** (Shutting Down) before the final transition to 8
2169 (Terminated).

2170 9.6.5 KillDiagnostic

2171 The client can immediately abort the execution of a test, with no attempt to perform a clean shutdown, by
2172 using the RequestStateChange() method call on the Job instance that is returned from the
2173 RunDiagnosticService() method, as shown in the following procedure. Assume that the client starts at a
2174 known DiagnosticTest instance.

2175 1) The client follows the ElementCapabilities association from the DiagnosticTest to the
2176 DiagnosticServiceJobCapabilities for the service.

2177 If no DiagnosticServiceJobCapabilities is returned, proceed to step 3. Support for Kill is
2178 mandatory.

2179 2) The client checks the DiagnosticServiceJobCapabilities.RequestedStatesSupported property for
2180 the value of 5 (Kill).

2181 If the value exists, the Job supports kill.

2182 3) The client finds the appropriate Job instances. The GetSpecificDiagnosticExecutions use case
2183 (see 9.7.3) describes the necessary steps.

2184 4) The client calls the RequestStateChange() method, passing in a RequestedState value of 5
2185 (Kill).

2186 After the transition is completed successfully, the ConcreteJob that represents the test will set
2187 the value of the JobState property to 9 (Killed) and set the value of TimeOfLastStateChange to
2188 the current time.

2189 9.7 Discover diagnostic executions

2190 In the following use cases, the term *execution* refers to an instance of the ConcreteJob class created to
2191 control a diagnostic service that was started on a managed element. The job may be in any of the states
2192 represented by the JobState property value, not necessarily active and running.

2193 The CIM_ prefix has been omitted from the class names in the use cases for readability.

2194 9.7.1 GetAffectedMEs

2195 The client can find all of the managed elements that are affected by a diagnostic execution as follows.
2196 Assume that the client starts at a known DiagnosticTest instance.

2197 1) From the DiagnosticTest instance, the client calls the Associators operation by using
2198 OwningJobElement as the association class and ConcreteJob as the result class.

2199 The operation returns the ConcreteJob instances launched by the DiagnosticTest.

2200 2) For each ConcreteJob instance, the client calls the Associators operation by using
2201 AffectedJobElement as the association class and ManagedElement as the result class.

2202 The operation returns the ManagedElement instances that this DiagnosticTest affects.

2203 NOTE This use case depends on the optional AffectedJobElement association. If that association does not exist,
2204 this use case is invalid.

2205 **9.7.2 GetAllDiagnosticExecutionsForME**

2206 The client can find all of the diagnostic executions on a system for a managed element as follows.
2207 Assume that the client starts at a known ManagedElement instance.

2208 1) From the ManagedElement instance, the client calls the Associators operation by
2209 using AffectedJobElement as the association class.

2210 The operation returns the ConcreteJob instances launched against this ManagedElement.

2211 2) For each ConcreteJob instance, the client calls the AssociatorNames operation by using
2212 OwningJobElement as the association class and DiagnosticTest as the result class.

2213 The operation returns the instance paths to the DiagnosticTest instances that launched the
2214 ConcreteJob against this ManagedElement. Each ConcreteJob instance that is associated with
2215 a DiagnosticTest represents an execution of a diagnostic service on that ManagedElement.

2216 NOTE This use case depends on the optional AffectedJobElement association. If that association does not exist,
2217 this use case is invalid.

2218 **9.7.3 GetSpecificDiagnosticExecutions**

2219 The client can find all of the executions of a specific diagnostic as follows. Assume that the client starts at
2220 a known DiagnosticTest instance.

2221 1) From the DiagnosticTest instance, the client calls the Associators operation by
2222 using OwningJobElement as the association class.

2223 The operation returns the ConcreteJob instances launched by the DiagnosticTest. Each
2224 ConcreteJob instance represents an execution of that diagnostic service.

2225 **9.7.4 GetSpecificDiagnosticExecutionsForME**

2226 The client can find all of the executions of a specific diagnostic for a particular managed element by using
2227 either of the following methods:

- 2228 • starting at the known ManagedElement instance
- 2229 • starting at the known DiagnosticTest instance

2230 **9.7.4.1 Starting at the Managed Element**

2231 NOTE This use case depends on the optional AffectedJobElement association. If that association does not exist,
2232 this use case is invalid.

2233 Assume that the client starts at the known ManagedElement instance and knows the particular
2234 DiagnosticTest instance.

2235 1) From the ManagedElement instance, the client calls the Associators operation by
2236 using AffectedJobElement as the association class and ConcreteJob as the result class.

2237 The operation returns the ConcreteJob instances that are running against this
2238 ManagedElement.

2239 2) For each ConcreteJob instance, the client calls the AssociatorNames operation by using
2240 OwningJobElement as the association class and DiagnosticTest as the result class.

2241 The operation returns the instance paths to the DiagnosticTest instances that launched the
2242 ConcreteJob instances against this ManagedElement.

2243 3) For each DiagnosticTest instance path returned, the client determines if it is the instance path of
2244 the known DiagnosticTest instance.

2245 If the instance path matches, the ConcreteJob instance represents an execution of that
2246 diagnostic service on that ManagedElement.

2247 **9.7.4.2 Starting at the DiagnosticTest**

2248 NOTE This use case depends on the optional AffectedJobElement association. If that association does not exist,
2249 this use case is invalid.

2250 Assume that the client starts at the known DiagnosticTest instance and knows the particular
2251 ManagedElement instance.

2252 1) From the DiagnosticTest instance, the client calls the Associators operation by using
2253 OwingJobElement as the association class and ConcreteJob as the result class.

2254 The operation returns the ConcreteJob instances launched by the DiagnosticTest.

2255 2) For each ConcreteJob instance, the client calls the AssociatorNames operation by using
2256 AffectedJobElement as the association class and ManagedElement as the result class.

2257 The operation returns the instance paths to the ManagedElement instances against which this
2258 DiagnosticTest launched the ConcreteJob instances.

2259 3) For each ManagedElement instance path returned, the client determines if it is the instance
2260 path of the known ManagedElement instance.

2261 If the instance path matches, the ConcreteJob instance represents an execution of that
2262 diagnostic service on that ManagedElement.

2263 **9.8 Discover diagnostic results (In Progress and Final)**

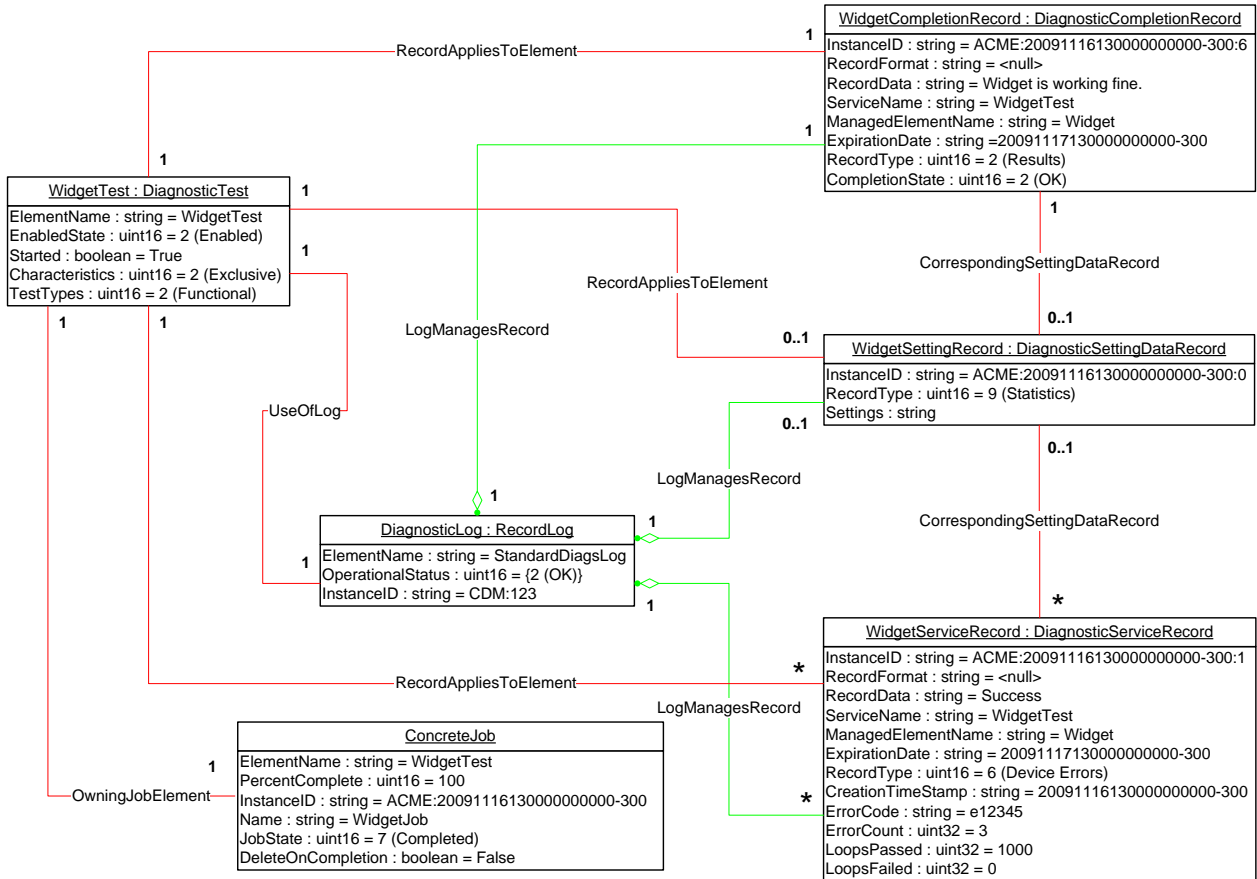
2264 In the following use cases, the term *execution* refers to an instance of the ConcreteJob class created to
2265 control a diagnostic service that was started on a managed element. The job may be in any of the states
2266 represented by the JobState property value, not necessarily active and running.

2267 Figure 5 is an object diagram that represents the results logging process for a diagnostic service on a
2268 fictitious device called "Widget". Only classes, properties, and methods that are of particular interest for
2269 the diagnostic model are shown.

2270 Figure 5 shows the required logging implementation, using the DiagnosticLog class. DiagnosticLog is a
2271 special subclass of RecordLog that supports a standard mechanism for organizing and retrieving (using
2272 ExecQuery) the records that diagnostics services generate. Use of this common logging mechanism can
2273 substantially increase interoperability and simplify client design.

2274 NOTE A separate DiagnosticLog instance shall be created each time the RunDiagnosticService method of
2275 DiagnosticTest is invoked.

2276 The CIM_ prefix has been omitted from the class names in the diagram and use cases for readability.



2277
2278

2279

Figure 5 – Diagnostic logging object diagram

2280 **9.8.1 GetLogRecordsForDiagnostic**

2281 The client can find all of the diagnostic log records for a particular diagnostic as follows. Assume that the
 2282 client starts at the known DiagnosticTest instance and that the DiagnosticRecord.ServiceName property
 2283 is implemented according to this profile.

2284 1) The client calls the ExecQuery operation as follows:

```

2285 SELECT * FROM CIM_DiagnosticRecord
2286 WHERE ServiceName = '<DiagnosticTest.Name>'
    
```

2287 The operation returns the DiagnosticRecord instances created for the specific DiagnosticTest,
 2288 independently if they are related to different managed elements or executions.

2289 An alternate method without using ExecQuery is as follows:

2290 Assume that the client starts at the known DiagnosticTest instance.

2291 1) From the DiagnosticTest instance, the client calls the Associators operation by using UseOfLog
 2292 as the association class and DiagnosticsLog as the result class.

2293 The operation returns the DiagnosticLog instances that contain records for the DiagnosticTest.

2294 2) For each DiagnosticLog instance, the client calls the Associators operation by using
 2295 LogManagesRecord as the association class and DiagnosticRecord as the result class.

2296 The operation returns the DiagnosticRecord instances in the log.

2297 3) For each returned instance, the client compares DiagnosticRecord.ServiceName with
2298 DiagnosticTest.Name to determine whether the instance is one created for the specific
2299 DiagnosticTest.

2300 **9.8.2 GetLogRecordsForME**

2301 The client can find all of the diagnostic log records for a particular managed element as follows. Assume
2302 that the client starts at the known ManagedElement instance and that the
2303 DiagnosticRecord.ManagedElementName property is implemented according to this profile.

2304 1) The client calls the ExecQuery operation as follows:

```
2305 SELECT * FROM CIM_DiagnosticRecord
2306 WHERE ManagedElementName = '<ManagedElement.ElementName>'
```

2307 The operation returns the DiagnosticRecord instances created for the specific
2308 ManagedElement, independently if they are related to different diagnostics or executions.

2309 An alternate method without using ExecQuery is as follows:

2310 Assume that the client starts at the known ManagedElement instance.

2311 1) From the ManagedElement instance, the client calls the Associators operation by using
2312 ServiceAvailableToElement as the association class and DiagnosticTest as the result class.

2313 The operation returns the DiagnosticTest instances for the ManagedElement.

2314 2) For each DiagnosticTest instance, the client calls the Associators operation by using UseOfLog
2315 as the association class and DiagnosticLog as the result class.

2316 The operation returns the DiagnosticLog instances that contain records for the DiagnosticTest.

2317 3) For each DiagnosticLog instance, the client calls the Associators operation by using
2318 LogManagesRecord as the association class and DiagnosticRecord as the result class.

2319 The operation returns the DiagnosticRecord instances in the log.

2320 4) For each returned instance, the client compares DiagnosticRecord.ManagedElementName with
2321 ManagedElement.ElementName to determine whether the instance is one created for the
2322 specific ManagedElement.

2323 **9.8.3 GetLogRecordsForMEAndDiagnostic**

2324 The client can find all of the diagnostic log records for a particular diagnostic run on a particular managed
2325 element as follows.

2326 Assume that the client starts at the known DiagnosticTest and ManagedElement instances and that the
2327 DiagnosticRecord.ServiceName and DiagnosticRecord.ManagedElementName properties are
2328 implemented according to this profile.

2329 1) The client calls the ExecQuery operation as follows:

```
2330 SELECT * FROM CIM_DiagnosticRecord
2331 WHERE ManagedElementName='<ManagedElement.ElementName>' and
2332 ServiceName='<DiagnosticTest.Name>'
```

2333 The operation returns the DiagnosticRecord instances created for the specific ManagedElement
2334 and DiagnosticTest, independently if they were created in different executions.

2335 An alternate method without using ExecQuery is as follows:

2336 Assume that the client starts at the known DiagnosticTest instance.

2337 1) From the DiagnosticTest instance, the client calls the Associators operation by using UseOfLog
2338 as the association class and DiagnosticLog as the result class.

2339 The operation returns the DiagnosticLog instances that contain records for the DiagnosticTest.

2340 2) For each DiagnosticLog instance, the client calls the Associators operation by using
2341 LogManagesRecord as the association class and DiagnosticRecord as the result class.

2342 The operation returns the DiagnosticRecord instances in the Log.

2343 3) For each returned instance, the client compares DiagnosticRecord.ServiceName with
2344 DiagnosticTest.Name and DiagnosticRecord.ManagedElementName with
2345 ManagedElement.ElementName to determine whether the instance is one created for the
2346 specific DiagnosticTest and ManagedElement.

2347 **9.8.4 GetDiagnosticExecutionFinalResults**

2348 The client can determine the final result of a diagnostic as follows. Assume that the client starts at the
2349 known ConcreteJob instance and that the DiagnosticRecord.InstanceID property follows the format
2350 defined in this profile (CIM_DiagnosticRecord.InstanceID *should* be <ConcreteJob.InstanceID>:<n>).
2351 This use case is also applicable after the job is completed and removed if the client knows the original
2352 ConcreteJob.InstanceID.

2353 1) The client calls the ExecQuery operation as follows:

```
2354 SELECT * FROM CIM_DiagnosticCompletionRecord
2355 WHERE InstanceID LIKE '<ConcreteJob.InstanceID>%'
```

2356 The operation returns the DiagnosticCompletionRecord instance created for the specific
2357 ConcreteJob.

2358 NOTE Only one DiagnosticCompletionRecord shall be returned.

2359 2) The client reads the DiagnosticCompletionRecord.CompletionState property, which shows the
2360 final result (Passed, Warning, Failed, Aborted, Incomplete, and so on) of the diagnostic
2361 execution.

2362 An alternate method without using ExecQuery is as follows:

2363 Assume that the client starts at the known DiagnosticTest instance.

2364 1) From the DiagnosticTest instance, the client calls the Associators operation by using UseOfLog
2365 as the association class and DiagnosticLog as the result class.

2366 The operation returns the DiagnosticLog instances that contain records for the DiagnosticTest.

2367 2) For each DiagnosticLog instance, the client calls the Associators operation by using
2368 LogManagesRecord as the association class and DiagnosticCompletionRecord as the result
2369 class.

2370 The operation returns the DiagnosticCompletionRecord instances in the Log.

2371 3) For each returned instance, the client compares DiagnosticCompletionRecord.ServiceName
2372 with DiagnosticTest.Name and DiagnosticRecord.ManagedElementName with
2373 ManagedElement.ElementName to determine whether the instance is one created for the
2374 specific DiagnosticTest and ManagedElement.

2375 **9.8.5 GetDiagnosticExecutionResults**

2376 The client can find all diagnostic log records for a particular execution (job) as follows.

2377 The diagnostic implementation will store the results of running the diagnostic in the manner selected
 2378 through the LogStorage setting. The most common mechanism is for the implementation to create
 2379 instances of DiagnosticRecord to record the results and status of running diagnostic services.
 2380 DiagnosticRecord has two subclasses: DiagnosticServiceRecord for recording test results, and
 2381 DiagnosticSettingDataRecord for preserving the test settings. The implementations for these classes will
 2382 implement ExecQuery to simplify the retrieval of records.

2383 The records are aggregated to a log by the LogManagesRecord association.

2384 Assume that the client starts at the known ConcreteJob instance and that the
 2385 DiagnosticRecord.InstanceID property follows the format defined in this profile
 2386 (CIM_DiagnosticRecord.InstanceID *should* be <ConcreteJob.InstanceID>:<n>). This use case is also
 2387 applicable after the job is completed and removed if the client knows the original ConcreteJob.InstanceID.

2388 1) The client calls the ExecQuery operation as follows:

```
2389     SELECT * FROM CIM_DiagnosticRecord
2390     WHERE InstanceID LIKE '<ConcreteJob.InstanceID>%'
```

2391 The operation returns the DiagnosticRecord instances created for the specific ConcreteJob
 2392 which may either be DiagnosticServiceRecord or DiagnosticSettingDataRecord instances.

2393 NOTE Only one DiagnosticSettingDataRecord shall be returned, while one or more DiagnosticServiceRecord
 2394 instances may be returned.

2395 **9.8.6 GetDiagnosticExecutionSettings**

2396 The client can find the settings used to execute a diagnostic as follows.

2397 Assume that the client starts at the known ConcreteJob instance and that the
 2398 DiagnosticRecord.InstanceID property follows the format defined in this profile
 2399 (CIM_DiagnosticRecord.InstanceID *should* be <ConcreteJob.InstanceID>:<n>). This use case is also
 2400 applicable after the job is completed and removed if the client knows the original ConcreteJob.InstanceID.

2401 1) The client calls the ExecQuery operation as follows:

```
2402     SELECT * FROM CIM_DiagnosticSettingDataRecord
2403     WHERE InstanceID LIKE '<ConcreteJob.InstanceID>%'
```

2404 The operation returns the DiagnosticSettingDataRecord instance created for the specific
 2405 ConcreteJob.

2406 NOTE Only one DiagnosticSettingDataRecord instance shall be returned.

2407 2) The client reads the DiagnosticSettingDataRecord.Settings property, which is a
 2408 DiagnosticSettingData embedded instance that contains the settings of the diagnostic
 2409 execution.

2410 An alternate method without using ExecQuery is as follows:

2411 Assume that the client starts at the known DiagnosticTest instance.

2412 1) From the DiagnosticTest instance, the client calls the Associators operation by using UseOfLog
 2413 as the association class and DiagnosticsLog as the result class.

2414 The operation returns the DiagnosticsLog instances that contain records for the DiagnosticTest.

2415 2) For each DiagnosticsLog instance, the client calls the Associators operation by using
 2416 LogManagesRecord as the association class and DiagnosticSettingDataRecord as the result
 2417 class.

2418 The operation returns the DiagnosticSettingDataRecord instances in the Log.

2419 3) For each returned instance, the client compares portion of DiagnosticRecord.InstanceID that
 2420 contains the ConcreteJob.InstanceID with ConcreteJob.InstanceID to determine whether the
 2421 instance is one created for the specific execution of the DiagnosticTest.

2422 4) The client reads the DiagnosticSettingDataRecord.Settings property, which is a
 2423 DiagnosticSettingData embedded instance that contains the settings of the diagnostic
 2424 execution.

2425 Another alternate method without using ExecQuery is as follows:

2426 NOTE This alternative use case depends on the implementation of DiagnosticSettingRecord and
 2427 CorrespondingSettingsRecord.

2428 Assume that the client starts at the known DiagnosticTest instance.

2429 1) From the DiagnosticTest instance, the client calls the Associators operation by using UseOfLog
 2430 as the association class and DiagnosticLog as the result class.

2431 The operation returns the DiagnosticLog instances that contain records for the DiagnosticTest.

2432 2) For each DiagnosticLog instance, the client calls the Associators operation by using
 2433 LogManagesRecord as the association class and DiagnosticSettingDataRecord as the result
 2434 class.

2435 The operation returns the DiagnosticSettingRecord instances in the Log.

2436 3) For each returned instance, the client compares portion of
 2437 DiagnosticSettingDataRecord.InstanceID with ConcreteJob.InstanceID to determine whether
 2438 the instance is the one created for the specific execution of the DiagnosticTest.

2439 4) From the DiagnosticSettingDataRecord instance, the client calls the Associators operation by
 2440 using CorrespondingSettingsRecord as the association class and DiagnosticServiceRecord as
 2441 the result class.

2442 The operation returns the DiagnosticServiceRecord instances created for the specific execution
 2443 of the DiagnosticTest

2444 9.8.7 GetDiagnosticProgress

2445 The client can get the progress of a running diagnostic as follows.

2446 The client may poll the ConcreteJob.PercentComplete property to determine test progress or register for
 2447 an indication that this property has changed. The value of this property shall be kept current to be useful.
 2448 Service implementations should update this property within one second of becoming aware of a progress
 2449 change.

2450 1) The client may use any of the Discover Diagnostic Execution use cases (see 9.7) to find the
 2451 desired ConcreteJob instances.

2452 2) The client reads the ConcreteJob.PercentComplete property to determine test progress.

2453 Assuming CIM_InstModification indications are supported, the client may register to receive indications
 2454 when the particular ConcreteJob.PercentComplete property changes value.

2455 1) The client can use any of the Discover Diagnostic Execution use cases (see 9.7) to find the
 2456 desired ConcreteJob instances.

2457 2) The client can register to receive a CIM_InstModification indication by creating an indication
 2458 subscription using the following CIM_IndicationFilter.Query:

```
2459 SELECT * FROM CIM_InstModification
2460 WHERE "SourceInstance.ISA("CIM_ConcreteJob") and
2461 SourceInstance.InstanceID=<ConcreteJob.InstanceID> and
2462 PreviousInstance.PercentComplete <> SourceInstance.PercentComplete
```


Element Name	Requirement	Description
CIM_ElementSoftwareIdentity	Mandatory	See 10.15.
CIM_FilterCollection	Optional	See 10.16
CIM_HelpService	Optional	See 10.16.
CIM_HostedService	Mandatory	See 10.18 and 9.1.
CIM_IndicationFilter	Mandatory	See 10.19
CIM_LogManagesRecord	Mandatory	See 10.20.
CIM_MemberOfCollection	Optional	See 10.21
CIM_OwningCollectionElement	Optional	See 10.22
CIM_RecordAppliesToElement	Optional	See 10.23.
CIM_RegisteredProfile	Mandatory	See 10.24.
CIM_ServiceAffectsElement	Mandatory	See 10.25.
CIM_ServiceAvailableToElement	Mandatory	See 10.26.
CIM_ServiceComponent	Optional	See 10.27.
CIM_SoftwareIdentity	Mandatory	See 10.28.
CIM_UseOfLog	Mandatory	See 10.29.
Indications		
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG0"	Mandatory	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG0" See 7.9.1
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG1"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG1" See 7.9.2
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG3"	Mandatory	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG3" See 7.9.3
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG4"	Mandatory	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG4" See 7.9.4
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG5"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG5" See 7.9.5
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG6"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG6" See 7.9.6
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG7"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG7" See 7.9.7
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG8"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG8" See 7.9.8

Element Name	Requirement	Description
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG10"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG10" See 7.9.9
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG11"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG11" See 7.9.10
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG13"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG13" See 7.9.11
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG14"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG14" See 7.9.12
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG15"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG15" See 7.9.13
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG16"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG16" See 7.9.14
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG17"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG17" See 7.9.15
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG18"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG18" See 7.9.16
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG22"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG22" See 7.9.17
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG23"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG23" See 7.9.18
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG24"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG24" See 7.9.19
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG26"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG26" See 7.9.20
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG27"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG27" See 7.9.21
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG28"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG28" See 7.9.22

Element Name	Requirement	Description
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG30"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG30" See 7.9.23
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG31"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG31" See 7.9.24
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG32"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG32" See 7.9.25
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG33"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG33" See 7.9.26
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG43"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG43" See 7.9.27
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG44"	Mandatory	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG44" See 7.9.28
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG45"	Mandatory	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG45" See 7.9.29
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG46"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG46" See 7.9.30
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="DMTF" and MessageID="DIAG47"	Optional	Query Language="DMTF:CQL" Name="DMTF:Diagnostics:DIAG47" See 7.9.31

2472 **10.1 CIM_AvailableDiagnosticService**

2473 CIM_AvailableDiagnosticService is used to discover the diagnostic services that are installed for a
 2474 particular managed element. Table 29 provides information about the properties of
 2475 CIM_AvailableDiagnosticService.

2476 **Table 29 – Class: CIM_AvailableDiagnosticService**

Properties	Requirement	Notes
ServiceProvided	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticService.
UserOfService	Mandatory	Key: This property shall be a reference to an instance of CIM_ManagedElement.
EstimatedDurationOfService	Mandatory	See 7.2.1.
EstimatedDurationQualifier	Optional	See 7.2.2.

2477 **10.2 CIM_CorrespondingSettingDataRecord (DiagnosticServiceRecord)**

2478 CIM_CorrespondingSettingDataRecord is used to associate a service record with the corresponding
 2479 setting data record. Table 30 provides information about the properties of
 2480 CIM_CorrespondingSettingDataRecord.

2481 **Table 30 – Class: CIM_CorrespondingSettingDataRecord**

Properties	Requirement	Notes
DataRecord	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticServiceRecord.
SettingsRecord	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticSettingDataRecord. Cardinality 1

2482 **10.3 CIM_CorrespondingSettingDataRecord (DiagnosticCompletionRecord)**

2483 CIM_CorrespondingSettingDataRecord is used to associate a completion record with the corresponding
 2484 setting data record. Table 31 provides information about the properties of
 2485 CIM_CorrespondingSettingDataRecord.

2486 **Table 31 – Class: CIM_CorrespondingSettingDataRecord**

Properties	Requirement	Notes
DataRecord	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticCompletionRecord.
SettingsRecord	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticSettingDataRecord. Cardinality 1

2487 **10.4 CIM_DiagnosticCompletionRecord**

2488 CIM_DiagnosticCompletionRecord is used to report the final state of diagnostic execution (OK, Failed,
 2489 Incomplete, Aborted, and so on). Table 32 provides information about the properties of
 2490 CIM_DiagnosticCompletionRecord.

2491 **Table 32 – Class: CIM_DiagnosticCompletionRecord**

Properties	Requirement	Notes
InstanceID	Mandatory	<p>Key:</p> <p>InstanceID should be constructed using the following preferred algorithm:</p> <p><ConcreteJob.InstanceID>:<n></p> <p>< ConcreteJob.InstanceID> is <OrgID>:<LocalID> as described in CIM_ConcreteJob, and <n> is an increment value that provides uniqueness. <n> should be set to \"0\" for the first record created by the job, and incremented for each subsequent record.</p> <p>(pattern "^.*[:].*[:][0123456789]*\$")</p>
CreationTimeStamp	Mandatory	None.
RecordData	Mandatory	None.
RecordFormat	Mandatory	None.
ServiceName	Mandatory	<p>The ServiceName property shall be constructed as follows: <OrgID>:<TestName>.</p> <p>(pattern "^.*[:].*\$")</p>
ManagedElementName	Mandatory	<p>This property will be formatted as a free-form string of variable length.</p> <p>(pattern ".*")</p>
RecordType	Mandatory	The record type shall be 2 (Results).
ExpirationDate	Mandatory	See 7.6.1.
CompletionState	Mandatory	None.
OtherCompletionStateDescription	Conditional	If CompletionState has the value 1 (Other), this property is Mandatory.
LoopsPassed	Optional	If looping is supported, this property is Mandatory.
LoopsFailed	Optional	If looping is supported, this property is Mandatory.
ErrorCode	Mandatory	<p>This property shall be an array that contains the error codes of all errors generated by the diagnostic service execution.</p> <p>If there are no errors, this property may have the value NULL.</p>

Properties	Requirement	Notes
ErrorCount	Mandatory	This property shall be an array where each position should contain the number of times that an error (which can be identified by the same position of the ErrorCode array) happened. If there are no errors, this property may have the value NULL.

2492 10.5 CIM_DiagnosticLog

2493 CIM_DiagnosticLog represents a log that aggregates all of the results (records) that the execution of a
2494 diagnostic generates. Table 33 provides information about the properties of CIM_DiagnosticLog.

2495 **Table 33 – Class: CIM_DiagnosticLog**

Properties	Requirement	Notes
InstanceID	Mandatory	Key: InstanceID should be constructed using the following preferred algorithm: <OrgID>:<LocalID> (See the MOF file for more detail.) (pattern "^.*[:].*\$")
ClearLog()	Mandatory	See 8.2.

2496 10.6 CIM_DiagnosticServiceCapabilities

2497 CIM_DiagnosticServiceCapabilities publishes the diagnostic service's capabilities, such as settings and
2498 execution controls that are supported. Table 34 provides information about the properties of
2499 CIM_DiagnosticServiceCapabilities.

2500 **Table 34 – Class: CIM_DiagnosticServiceCapabilities**

Properties	Requirement	Notes
InstanceID	Mandatory	Key: InstanceID shall be unique and should be constructed using the following preferred algorithm: <OrgID>:<LocalID> (See the MOF file for more detail.) <LocalID> should be set to the Name property value of the Service to which these capabilities apply. (pattern "^.*[:].*\$")
ElementName	Mandatory	This property shall contain the value of the Service's ElementName property. The property will be formatted as a free-form string of variable length. (pattern ".*")

Properties	Requirement	Notes
SupportedServiceModes	Optional	If service modes are supported, they shall be published using this property.
OtherSupportedServiceModesDescriptions	Conditional	If SupportedServiceModes includes the value of 1 (Other), this property is Mandatory.
SupportedLoopControl	Optional	If looping is supported, its controls shall be published using this property.
OtherSupportedLoopControlDescriptions	Conditional	If SupportedLoopControl includes the value 1 (Other), this property is Mandatory.
SupportedLogOptions	Optional	If any log options are supported, they shall be published using this property.
OtherSupportedLogOptionsDescriptions	Conditional	If SupportedLogOptions includes the value 1 (Other), this property is Mandatory.
SupportedLogStorage	Optional	If any log storage options are supported, they shall be published using this property.
OtherSupportedLogStorageDescriptions	Conditional	If SupportedLogStorage includes the value 1 (Other), this property is Mandatory.
SupportedExecutionControls	Optional	Deprecated: If any execution controls are supported, they shall be published using this property.
OtherSupportedExecutionControls Descriptions	Conditional	Deprecated: If SupportedExecutionControls includes the value 1 (Other), this property is Mandatory.

2501 **10.7 CIM_DiagnosticServiceRecord**

2502 CIM_DiagnosticServiceRecord is used to report diagnostic service messages, such as results, errors,
 2503 warnings, and status. Table 35 provides information about the properties of
 2504 CIM_DiagnosticServiceRecord.

2505 **Table 35 – Class: CIM_DiagnosticServiceRecord**

Properties	Requirement	Notes
InstanceID	Mandatory	Key: InstanceID should be constructed using the following preferred algorithm: <ConcreteJob.InstanceID>:<n> Where < ConcreteJob.InstanceID> is <OrgID>:<LocalID> as described in ConcreteJob and <n> is an increment value that provides uniqueness. <n> should be set to \"0\" for the first record created by the job, and incremented for each subsequent record. (pattern \"^.*[:].*[:][0123456789]*\$")
CreationTimeStamp	Mandatory	None.
RecordData	Mandatory	None.
RecordFormat	Mandatory	None.
LoopsPassed	Mandatory	None.

Properties	Requirement	Notes
LoopsFailed	Mandatory	None.
ErrorCode	Conditional	<p>If the RecordType value is 7(Device Errors) or 8 (Service Errors), this property shall be an array that contains only one error code number.</p> <p>If the RecordType value is 2 (Results), this property shall be an array that contains the error codes of all errors generated by the diagnostic service or subtest execution at the time when the record was logged.</p> <p>If the RecordType value is not 2 (Results) or 7(Device Errors) or 8 (Service Errors), this property may be NULL.</p> <p>The property will be formatted as a free-form string of variable length. (pattern ".*")</p>
ErrorCount	Conditional	<p>If the RecordType value is 7(Device Errors) or 8 (Service Errors), this property shall be an array that has just one element whose value is 1.</p> <p>If the RecordType value is 2 (Results), this property should be an array where each position should contain the number of times that an error occurred that can be identified by the same position in the ErrorCode array.</p> <p>If the RecordType value is not 2 (Results) or 7(Device Errors) or 8 (Service Errors), this property may be NULL.</p>
ServiceName	Mandatory	<p>This property shall be constructed as follows: <OrgID>:<TestName>.</p> <p>(pattern "^[[:].*\$")</p>
ManagedElementName	Mandatory	<p>This property shall be formatted as a free-form string of variable length.</p> <p>(pattern ".*")</p>
RecordType	Mandatory	<p>A RecordType value of 2 (Results) shall be used to log interim results from the diagnostic service execution (for example, results from a subtest).</p>
OtherRecordTypeDescription	Conditional	<p>If RecordType has the value 1 (Other), this property is Mandatory.</p>
ExpirationDate	Mandatory	See 7.6.1.

2506 **10.8 CIM_DiagnosticSettingData (Default)**

2507 Diagnostic services use CIM_DiagnosticSettingData to publish default settings by using
 2508 CIM_ElementSettingData where the IsDefault property has the value of TRUE. Table 36 provides
 2509 information about the properties of CIM_DiagnosticSettingData.

2510 **Table 36 – Class: CIM_DiagnosticSettingData**

Properties	Requirement	Notes
InstanceID	Mandatory	<p>Key:</p> <p>InstanceID should be constructed using the following preferred algorithm:</p> <p><OrgID>:<LocalID></p> <p>(See the MOF file for more detail.)</p> <p><LocalID> should be set to a time stamp (CIM DateTime).</p> <p>For example:</p> <p>ACME:19980525133015.0000000-300</p> <p>(pattern "^.*[:].*\$")</p>
ElementName	Mandatory	This property shall be formatted as a free-form string of variable length. (pattern ".*")
HaltOnError	Optional	<p>If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 4 (HaltOnError), this property can be used to affect test behavior.</p> <p>When this property is TRUE, the service should halt after finding the first error.</p>
QuickMode	Optional	<p>If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 3 (QuickMode), this property can be used to affect test behavior.</p> <p>When this property is TRUE, the service should attempt to run in an accelerated fashion either by reducing the coverage or by reducing the number of tests performed.</p>
PercentOfTestCoverage	Optional	<p>If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 2 (PercentOfTestCoverage), this property can be used to affect test behavior.</p> <p>This property requests that the service reduce test coverage to the specified percentage.</p>
NonDestructive	Optional	<p>If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 7 (NonDestructive), this property can be used to affect test behavior.</p> <p>When this property is TRUE, the service should not run destructive tests.</p>

Properties	Requirement	Notes
LoopControl	Optional	This property is used in combination with LoopControlParameter to set one or more loop control mechanisms that limit the number of times that a test should be repeated.
LoopControlParameter	Conditional	<p>If a LoopControl includes the value of 3 (Count) or 5 (ErrorCount), the corresponding LoopControlParameter array element shall represent a uint32 numeric value.</p> <p>If a LoopControl includes the value of 4 (Timer), the corresponding LoopControlParameter array element shall represent a datetime value.</p> <p>(pattern "<code>^b[01]* ^d[0123456789]* ^x[0123456789ABCDEFabcdef]* ^[0123456789]*</code>")</p>
OtherLoopControlDescriptions	Conditional	If LoopControl includes the value 1 (Other), this property is Mandatory.
ResultPersistence	Mandatory	<p>If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 5 (ResultPersistence), this property can be used to affect test behavior.</p> <p>This property specifies how many seconds the records should persist after service execution finishes. 0 (zero) indicates "no persistence" and 0xFFFFFFFF indicates "persist forever".</p> <p>See 7.6.1.</p>
LogOptions	Optional	This property specifies the types of data that should be logged by the diagnostic service.
OtherLogOptionsDescriptions	Conditional	If LogOptions includes the value 1 (Other), this property is Mandatory.
LogStorage	Optional	<p>This property specifies the logging mechanism to store the diagnostic results.</p> <p>This property must be one of the values in DiagnosticServiceCapabilities.LogStorage</p>
OtherLogStorageDescriptions	Conditional	If LogStorage includes the value 1 (Other), this property is Mandatory.
VerbosityLevel	Optional	This property specifies the desired volume or detail logged by a diagnostic service.

2511 **10.9 CIM_DiagnosticSettingData (Client)**

2512 A client uses CIM_DiagnosticSettingData to override the defaults settings and run a diagnostic service
 2513 using specific settings. Such settings are passed as the DiagnosticSettings argument when the
 2514 RunDiagnosticService() extrinsic method of CIM_DiagnosticTest is invoked. Table 37 provides
 2515 information about the properties of CIM_DiagnosticSettingData.

2516 **Table 37 – Class: CIM_DiagnosticSettingData**

Properties	Requirement	Notes
InstanceID	Mandatory	<p>Key:</p> <p>InstanceID should be constructed using the following preferred algorithm:</p> <p><OrgID>:<LocalID></p> <p>(See the MOF file for more detail.)</p> <p><LocalID> should be set to a time stamp (CIM DateTime).</p> <p>For example:</p> <p>ACME:19980525133015.0000000-300</p> <p>(pattern "^.*[:].*\$")</p>
ElementName	Mandatory	This property shall be formatted as a free-form string of variable length. (pattern ".*")
HaltOnError	Optional	<p>If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 4 (HaltOnError), this property can be used to affect test behavior.</p> <p>When this property is TRUE, the service should halt after finding the first error.</p>
QuickMode	Optional	<p>If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 3 (QuickMode), this property can be used to affect test behavior.</p> <p>When this property is TRUE, the service should attempt to run in an accelerated fashion either by reducing the coverage or by reducing the number of tests performed.</p>
PercentOfTestCoverage	Optional	<p>If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 2 (PercentOfTestCoverage), this property can be used to affect test behavior.</p> <p>This property requests that the service reduce test coverage to the specified percentage.</p>
NonDestructive	Optional	<p>If the DiagnosticServiceCapabilities.SupportedServiceModes includes a value of 7 (NonDestructive), this property can be used to affect test behavior.</p> <p>When this property is TRUE, the service should not run destructive tests.</p>

Properties	Requirement	Notes
LoopControl	Optional	This property is used in combination with LoopControlParameter to set one or more loop control mechanisms that limit the number of times that a test should be repeated.
LoopControlParameter	Conditional	<p>If a LoopControl includes the value of 3 (Count) or 5 (ErrorCount), the corresponding LoopControlParameter array element shall represent a uint32 numeric value.</p> <p>If a LoopControl includes the value of 4 (Timer), the corresponding LoopControlParameter array element shall represent a datetime value.</p> <p>(pattern "<code>^b[01]* ^d[0123456789]* ^x[0123456789ABCDEFabcdef]* ^[0123456789]*</code>")</p>
OtherLoopControlDescriptions	Conditional	If LoopControl includes the value 1 (Other), this property is Mandatory.
ResultPersistence	Mandatory	<p>If the DiagnosticServiceCapabilities.SupportedServiceModes array contains a value of 5 (ResultPersistence), this property can be used to affect test behavior.</p> <p>This property specifies how many seconds the records should persist after service execution finishes. 0 (zero) indicates "no persistence" and 0xFFFFFFFF indicates "persist forever". See 7.6.1.</p>
LogOptions	Optional	This property specifies the types of data that should be logged by the diagnostic service.
OtherLogOptionsDescriptions	Conditional	If LogOptions includes the value 1 (Other), this property is Mandatory.
LogStorage	Optional	<p>This property specifies the logging mechanism to store the diagnostic results.</p> <p>This property must be one of the values in DiagnosticServiceCapabilities.LogStorage</p>
OtherLogStorageDescriptions	Conditional	If LogStorage includes the value 1 (Other), this property is Mandatory.
VerbosityLevel	Optional	This property specifies the desired volume or detail logged by a diagnostic service.

2517 **10.10 CIM_DiagnosticSettingDataRecord**

2518 CIM_DiagnosticSettingDataRecord stores the settings used in a specific diagnostic service execution.

2519 Table 38 provides information about the properties of CIM_DiagnosticSettingDataRecord.

2520 **Table 38 – Class: CIM_DiagnosticSettingDataRecord**

Properties	Requirement	Notes
InstanceID	Mandatory	<p>Key:</p> <p>InstanceID should be constructed using the following preferred algorithm:</p> <p><ConcreteJob.InstanceID>:<n></p> <p>< ConcreteJob.InstanceID> is <OrgID>:<LocalID> as described in CIM_ConcreteJob, and <n> is an increment value that provides uniqueness. <n> should be set to \"0\" for the first record created by the job, and incremented for each subsequent record.</p> <p>(pattern "^. *[:]. *:[0123456789]*\$")</p>
CreationTimeStamp	Mandatory	None.
ServiceName	Mandatory	<p>This property shall be constructed as follows:</p> <p><OrgID>:<TestName>.</p> <p>(pattern "^. *[:]. *\$")</p>
ManagedElementName	Mandatory	<p>This property will be formatted as a free-form string of variable length.</p> <p>(pattern ". *")</p>
RecordType	Mandatory	A RecordType value of 9 (Results) shall be used to log a DiagnosticSettingDataRecord.
ExpirationDate	Mandatory	See 7.6.1.
Settings	Conditional	<p>This property is set to a string that encodes a DiagnosticSettingData instance.</p> <p>If an instance of CIM_DiagnosticSettingData is associated through CIM_ElementSettingData to the instance of CIM_DiagnosticTest at the time the Diagnostic Service is run, this property is Mandatory.</p>

2521 **10.11 CIM_DiagnosticTest**

2522 CIM_DiagnosticTest is a class that represents a diagnostic service developed to exercise and observe
 2523 the behavior of a device that is implicated in some level of system malfunction. It contains properties
 2524 useful in test configuration and the RunDiagnosticService() method, a standard mechanism for invoking
 2525 the test.

2526 Table 39 provides information about the properties of CIM_DiagnosticTest.

2527 **Table 39 – Class: CIM_DiagnosticTest**

Properties	Requirement	Notes
SystemCreationClassName	Mandatory	Key
SystemName	Mandatory	Key
CreationClassName	Mandatory	Key
Name	Mandatory	Key: The Name property shall be constructed as follows: <OrgID>:<TestName>. (pattern "^.*[:].*\$")
ElementName	Mandatory	The property will be formatted as a free-form string of variable length. (pattern ".*")
Characteristics	Mandatory	See 7.1.3.
OtherCharacteristicsDescriptions	Conditional	If Characteristics includes the value of 1 (Other), this property is Mandatory.
TestTypes	Optional	See 7.1.4
OtherTestTypesDescriptions	Optional	See 7.1.5
RunDiagnosticService()	Mandatory	See 8.1.

2528 **10.12 CIM_ElementCapabilities**

2529 CIM_ElementCapabilities associates a diagnostic service with its capabilities. Table 40 provides
 2530 information about the properties of CIM_ElementCapabilities.

2531 **Table 40 – Class: CIM_ElementCapabilities**

Properties	Requirement	Notes
ManagedElement	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticService.
Capabilities	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticServiceCapabilities. Cardinality 0..1

2532 **10.13 CIM_ElementSettingData (JobSettingData)**

2533 CIM_ElementSettingData associates the job settings with the job used to run a diagnostic test. Table 41
 2534 provides information about the properties of CIM_ElementSettingData.

2535

Table 41 – Class: CIM_ElementSettingData

Properties	Requirement	Notes
ManagedElement	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticService. Cardinality 1
SettingData	Mandatory	Key: This property shall be a reference to an instance of CIM_JobSettingData. Cardinality 0..1
IsDefault	Mandatory	If the instance of CIM_JobSettingData is the default setting, this property shall have the value of TRUE. Otherwise, this property shall have the value of FALSE.

2536 **10.14 CIM_ElementSettingData (DiagnosticSettingData)**

2537 CIM_ElementSettingData associates the diagnostic service with its default. Table 42 provides information
2538 about the properties of CIM_ElementSettingData.

2539

Table 42 – Class: CIM_ElementSettingData

Properties	Requirement	Notes
ManagedElement	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticService. Cardinality 1
SettingData	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticSettingData. Cardinality 0..1
IsDefault	Mandatory	If the instance of CIM_DiagnosticSettingData is the default setting, this property shall have the value of TRUE. Otherwise, this property shall have the value of FALSE.

2540 **10.15 CIM_ElementSoftwareIdentity**

2541 CIM_ElementSoftwareIdentity associates the diagnostic service with its version information. Table 43
2542 provides information about the properties of CIM_ElementSoftwareIdentity.

2543

Table 43 – Class: CIM_ElementSoftwareIdentity

Properties	Requirement	Notes
Antecedent	Mandatory	Key: This property shall be a reference to an instance of CIM_SoftwareIdentity. Cardinality 1.

Properties	Requirement	Notes
Dependent	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticService. Cardinality 1.

2544 10.16 CIM_FilterCollection

2545 CIM_FilterCollection represents a ProfileSpecificFilterCollection as defined in [DSP1054](#). It defines the
2546 collection of all the alert indications of the Diagnostics profile. Table 44 contains the requirements for
2547 elements of this class.

2548 **Table 44 - Class: CIM_FilterCollection**

Properties	Requirement	Notes
InstanceID	Mandatory	Key: See DSP1054 .
CollectionName	Mandatory	The property shall be "DMTF:Diagnostics:ProfileSpecifiedAlertIndicationFilterCollection".

2549 10.17 CIM_HelpService

2550 CIM_HelpService is the preferred way for a service to publish online help information. Table 45 provides
2551 information about the properties of CIM_HelpService.

2552 **Table 45 – Class: CIM_HelpService**

Properties	Requirement	Notes
SystemCreationClassName	Mandatory	Key
SystemName	Mandatory	Key
CreationClassName	Mandatory	Key
Name	Mandatory	Key: This property will be formatted as a free-form string of variable length. (pattern ".**")
ElementName	Mandatory	This property will be formatted as a free-form string of variable length. (pattern ".**")
DeliveryOptions	Mandatory	None.
OtherDeliveryOptionDescription	Conditional	If DeliveryOptions has the value of 1 (Other), this property is Mandatory.
DocumentsAvailable	Mandatory	This property will be formatted as a free-form string of variable length. (pattern ".**")
DocumentDescriptions	Mandatory	None.
DocumentFormat	Mandatory	None.
OtherDocumentFormatDescription	Conditional	If DocumentFormat has the value of 1 (Other), this property is Mandatory.
GetHelp()	Mandatory	See 8.3.

2553 **10.18 CIM_HostedService**

2554 CIM_HostedService is used to associate an instance of CIM_DiagnosticTest with an instance of
 2555 CIM_ComputerSystem to which the CIM_DiagnosticTest is scoped and to associate an instance of
 2556 CIM_HelpService with an instance of CIM_ComputerSystem to which the CIM_HelpService is scoped.
 2557 Table 46 provides information about the properties of CIM_HostedService.

2558 **Table 46 – Class: CIM_HostedService**

Properties	Requirement	Notes
Antecedent	Mandatory	Key: This property shall be a reference to an instance of CIM_ComputerSystem. Cardinality 1
Dependent	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticTest. Cardinality 1..*

2559 **10.19 CIM_IndicationFilter**

2560 CIM_IndicationFilter represents a StaticIndicationFilter as defined in [DSP1054](#). It defines the format of all
 2561 the alert indication filters of the Diagnostics profile. Table 47 contains the requirements for elements of
 2562 this class.

2563 **Table 47 - Class: CIM_IndicationFilter**

Properties	Requirement	Notes
Name	Mandatory	Key: See the Name values as identified in Table 28.
CreationClassName	Mandatory	Key: See DSP1054 .
SystemName	Mandatory	Key: See DSP1054 .
SystemCreationClassName	Mandatory	Key: See DSP1054 .
SourceNamespaces[]	Mandatory	See DSP1054 .
IndividualSubscriptionSupported	Mandatory	See DSP1054 .
Query	Mandatory	See the Query values as identified in Table 28.
QueryLanguage	Mandatory	See the QueryLanguage values as identified in Table 28.

2564

2565 **10.20 CIM_LogManagesRecord**

2566 CIM_LogManagesRecord associates a log with its records (service records, setting records, or
 2567 completion records). Table 48 provides information about the properties of CIM_LogManagesRecord.

2568 **Table 48 – Class: CIM_LogManagesRecord**

Properties	Requirement	Notes
Log	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticLog.
Record	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticRecord.

2569 **10.21 CIM_MemberOfCollection**

2570 CIM_MemberOfCollection represents an association between the profile specific FilterCollection and the
 2571 CIM_IndicationFilters for the alert indications. Table 49 contains the requirements for elements of this
 2572 class.

2573 **Table 49 - Class: CIM_MemberOfCollection**

Properties	Requirement	Notes
Collection	Mandatory	Key: Value shall reference the profile specific FilterCollection instance representing a filter collection containing the alert indication filters.
Member	Mandatory	Key: Value shall reference an Alert IndicationFilter instance representing a contained alert indication filter.

2574

2575 **10.22 CIM_OwningCollectionElement**

2576 CIM_OwningCollectionElement represents an association between the IndicationService that controls the
 2577 profile specific FilterCollection and the profile specific CIM_FilterCollection for the alert indication filters.
 2578 Table 50 contains the requirements for elements of this class.

2579 **Table 50 - Class: CIM_OwningCollectionElement**

Properties	Requirement	Notes
OwningElement	Mandatory	Key: See DSP1054 .
OwnedElement	Mandatory	Key: Value shall reference the profile specific Alert Indication FilterCollection instance

2580

2581 **10.23 CIM_RecordAppliesToElement**

2582 CIM_RecordAppliesToElement associates a record with the managed elements (diagnostic service and
 2583 device) that have a relationship with this record. Table 51 provides information about the properties of
 2584 CIM_RecordAppliesToElement.

2585 **Table 51 – Class: CIM_RecordAppliesToElement**

Properties	Requirement	Notes
Antecedent	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticRecord.
Dependent	Mandatory	Key: This property shall be a reference to an instance of CIM_ManagedElement.

2586 **10.24 CIM_RegisteredProfile**

2587 CIM_RegisteredProfile identifies the *Diagnostics Profile* in order for a client to determine whether an
 2588 instance of CIM_DiagnosticService is conformant with this profile. The CIM_RegisteredProfile class is
 2589 defined by [DSP1033 Profile Registration Profile](#). With the exception of the mandatory values specified in
 2590 Table 52, the behavior of the CIM_RegisteredProfile instance is in accordance with DSP1033 [DSP1033](#)
 2591 [Profile Registration Profile](#).

2592 **Table 52 – Class: CIM_RegisteredProfile**

Properties	Requirement	Notes
RegisteredName	Mandatory	This property shall have a value of "Diagnostics".
RegisteredVersion	Mandatory	This property shall have a value of "2.0.0".
RegisteredOrganization	Mandatory	This property shall have a value of 2 (DMTF).

2593 **10.25 CIM_ServiceAffectsElement**

2594 CIM_ServiceAffectsElement is used to associate to the diagnostic service any managed elements that
 2595 are affected by the running of the service. Table 53 provides information about the properties of
 2596 CIM_ServiceAffectsElement.

2597 **Table 53 – Class: CIM_ServiceAffectsElement**

Properties	Requirement	Notes
AffectedElement	Mandatory	Key: This property shall be a reference to an instance of CIM_ManagedElement.
AffectingElement	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticService.

2598 **10.26 CIM_ServiceAvailableToElement**

2599 CIM_ServiceAvailableToElement associates the diagnostic service with its help service information. Table
 2600 54 provides information about the properties of CIM_ServiceAvailableToElement.

2601 **Table 54 – Class: CIM_ServiceAvailableToElement**

Properties	Requirement	Notes
ServiceProvided	Mandatory	Key: This property shall be a reference to an instance of CIM_HelpService. Cardinality 1
UserOfService	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticService. Cardinality 1

2602 **10.27 CIM_ServiceComponent**

2603 CIM_ServiceComponent associates a test that is also part of another test. This class is used when
 2604 DiagnosticTest.Characteristics includes the value 6 (Is Package) and subtests are implemented as
 2605 separate instances of DiagnosticTest. Table 55 provides information about the properties of
 2606 CIM_ServiceComponent.

2607 **Table 55 – Class: CIM_ServiceComponent**

Properties	Requirement	Notes
GroupComponent	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticService.
PartComponent	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticService.

2608 **10.28 CIM_SoftwareIdentity**

2609 CIM_SoftwareIdentity is used to publish version information about the diagnostic service. Table 56
 2610 provides information about the properties of CIM_SoftwareIdentity.

2611 **Table 56 – Class: CIM_SoftwareIdentity**

Properties	Requirement	Notes
InstanceID	Mandatory	Key: InstanceID should be constructed using the following preferred algorithm: <OrgID>:<LocalID> (See the MOF file for more detail.) (pattern "^.*[:].*\$")
MajorVersion	Mandatory	None.
MinorVersion	Mandatory	None.
RevisionNumber	Mandatory	None.
VersionString	Mandatory	None.
Manufacturer	Mandatory	This property will be formatted as a free-form string of variable length. (pattern ".*")

2612 **10.29 CIM_UseOfLog**

2613 CIM_UseOfLog associates a log with a managed element (a device or diagnostic service) whose
 2614 information is stored in the log. Table 57 provides information about the properties of CIM_UseOfLog.

2615 **Table 57 – Class: CIM_UseOfLog**

Properties	Requirement	Notes
Antecedent	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticLog.
Dependent	Mandatory	Key: This property shall be a reference to an instance of CIM_DiagnosticService.

2616
2617
2618
2619

ANNEX A (informative)

Change log

Version	Date	Description
1.0.0a	2006-04-17	Preliminary
1.0.1	2009-09-23	Final Standard
2.0.0	2010-08-13	DMTF Draft Standard
2.1.0	2013-05-09	<p>Changed the version</p> <p>Changed the Date</p> <p>Changed the Document Status</p> <p>Edited the Normative References</p> <p>Added Diagnostic Job Control and Indications to the Related Profile table of the Synopsis</p> <p>Clause 7 changes</p> <ul style="list-style-type: none"> - Added a clause "7.1.4 CIM_DiagnosticTest.TestType" to define TestType - Greatly expanded clause "7.3 CIM_DiagnosticServiceCapabilities" - Greatly expanded clause 7.4 "CIM_DiagnosticServiceSettingData" - Deleted clause "7.5 CIM_ConcreteJob" (it's moved to DSP1119) - Added a clause "7.8 Diagnostics Profile Indications Support" - Added a clause "7.9 Diagnostics Alert Indications and Standard Messages" <p>Clause 8 Methods</p> <ul style="list-style-type: none"> - Removed references to CreateInstance, since it left too many questions unanswered <p>Clause 9 Use Cases</p> <ul style="list-style-type: none"> - Minor editing of the use cases <p>Clause 10 CIM Elements changes</p> <ul style="list-style-type: none"> - Deleted the classes moved to Diagnostic Job Control (CIM_AffectedJobElement, CIM_ConcreteJob, CIM_JobSettingData and CIM_OwningJobElement) - Added the entries for the Alert Indications
2.1.0a	2013-06-13	Released as Work in Progress

2620

Bibliography

2621 DMTF DSP2000, *CIM Diagnostic Model White Paper 1.0*,
2622 http://www.dmtf.org/standards/published_documents/DSP2000.pdf

2623

2624